Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matej Hočevar

Implementacija spletne trgovine in priporočilnega sistema

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Aleš Smrdel

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomski nalogi razvijte aplikacijo za spletno trgovino. Pri razvoju aplikacije za spletno trgovino najprej analizirajte zahteve različnih skupin uporabnikov, kot so na primer stranke ali pa upravitelji. Nato izdelajte načrt aplikacije znotraj katerega tudi definirajte arhitekturo ter podatkovni model primeren za spletno trgovino. Pri izdelavi same aplikacije pa implementirajte funkcionalnosti za različne skupine uporabnikov ugotovljene v prvem koraku. Pri tem izberite in uporabite najprimernejše tehnologije na strani strežnika in odjemalca. Ker je pri tovrstnih aplikacijah velika verjetnost različnih zlorab, naj bo aplikacija implementirana tako, da omogoča čim boljšo obrambo pred različnimi možnimi zlorabami. Za zagotavljanje čim boljše uporabniške izkušnje pa implementirajte tudi priporočilni sistem, ki bo uporabniku priporočal izdelke glede na uporabnikove predhodne nakupe v spletni trgovini. Pri implementaciji odjemalskega dela pa posvetite pozornost tudi primernemu prikazu uporabniškega vmesnika aplikacije na različnih velikostih zaslonov.

Zahvaljujem se mentorju doc. dr. Alešu Smrdelu za strokovno pomoč in vodenje pri izdelavi diplomskega dela. Posebej se zahvaljujem zaročenki, družini in prijateljem, ki so me ves čas študija podpirali, spodbujali in mi stali ob strani. Zahvala gre tudi Urošu Plazniku, ki mi je v okviru svojega podjetja ponudil možnost izdelave diplomskega dela.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
HTML	HyperText Markup Language	jezik za označevanje nadbesedila
CSS	Cascading Style Sheets	kaskadne stilske predloge
JSON	JavaScript Object Notation	standard za izmenjavo podatkov
XML	eXtensible Markup Language	razširljiv označevalni jezik
CSV	Comma-Separated Values	format datoteke z vrednostmi, ločenimi z vejico
MVC	Model - View - Controller	Model - Pogled - Kontroler
DRY	Don't Repeat Yourself	princip "Ne ponavljaj se"
API	Application Programming Interface	aplikacijski programski vmesnik
HTTP	Hypertext Transfer Protocol	protokol za prenos nadbesedila
SQL	Structured Query Language	strukturiran povpraševalni jezik
XSS	Cross Site Scripting	večdomensko izvajanje kode
CSRF	Cross Site Request Forgery	večdomensko ponarejanje zahtevka

Povzetek

Naslov: Implementacija spletne trgovine in priporočilnega sistema

Avtor: Matej Hočevar

V okviru diplomskega dela je bila z uporabo modernih tehnologij najprej razvita spletna trgovina. Pri razvoju je bila pozornost usmerjena tudi na izredno pomemben segment naprav, ki jih uporabniki uporabljajo, to so mobilne naprave. Poleg tega pa je bil kot pomoč kupcem pri izbiri in naročanju izdelkov v spletni trgovini razvit tudi priporočilni sistem. V prvem delu diplomskega dela je predstavljeno, kaj so priporočilni sistemi, podane so tudi prednosti njihove uporabe in kako jih spletni trgovci uporabljajo pri izboljševanju prodaje na spletu. Poleg tehnologij, uporabljenih za implementacijo, bo bralec v drugem delu spoznal tudi uporabniške zahteve ter procese in funkcionalnosti, ki jih moderna spletna trgovina ponuja. V tretjem delu je predstavljena implementacija priporočilnih sistemov. Priporočilni sistemi so prikazani v praksi, na primeru spletne trgovine, ki je bila implementirana v okviru diplomske naloge. Na koncu sledita še analiza implementiranega priporočilnega sistema ter sklepne ugotovitve.

Ključne besede: priporočilni sistem, spletna trgovina, implementacija spletne aplikacije, prodaja.

Abstract

Title: Implementation of e-commerce site and recommendation system

Author: Matej Hočevar

In the scope of the diploma thesis the e-commerce site was developed first using modern technologies. While developing the site we concentrated on one of the most important devices people use every day - mobile phones. A recommendation system was also developed to help online customers select and order products. In the first part of the thesis recommendation systems are presented to explain how users can benefit from using them and how online sellers use them in order to improve sales. In addition to the technologies which were used for implementation, the reader will get familiar with user requirements and also processes and functionalities that an e-commerce site has to offer. In the third part of the thesis the implementation of recommendation systems is presented. These systems are also shown in practice through the e-commerce site which was implemented in the scope of this thesis. The last part of the thesis is the analysis of the implemented recommendation system and final conclusions.

Keywords: recommendation system, e-commerce, implementation of web application, sale.

Poglavje 1

Uvod

Z vse večjo uporabo informacijskih tehnologij v vsakdanjem življenju se spreminjajo tudi navade ljudi. Kar je bilo še pred nekaj leti bolj kot ne posebnost je lahko sedaj nekaj povsem vsakdanjega oziroma samoumevnega. Tako postajajo tudi spletne trgovine nekaj povsem običajnega, kupci oziroma uporabniki pa se jih poslužujejo vsakodnevno. Ena izmed prednosti, ki jih nudijo spletne trgovine je ta, da imamo z obiskom spletne trgovine na voljo več tisoč različnih izdelkov, med katerimi lahko izbiramo. Z vse večjo količino izdelkov v ponudbi spletnih trgovcev moramo potrošniki predelati vse več informacij preden izberemo izdelek, ki ustreza našim željam. Ena od rešitev, ki olajšujejo procesiranje informacij o želenem izdelku, so priporočilni sistemi [14].

1.1 Cilji diplomske naloge

Ciljev diplomske naloge je več. Prvi cilj v okviru diplomske naloge je seznanitev s priporočilnimi sistemi in kako jih velika spletna prodajna mesta uporabljajo pri svojem delovanju. Naslednji cilji so izdelava spletne trgovine z uporabo modernih tehnologij, implementacija priporočilnih sistemov in še analiza implementiranih priporočilnih sistemov.

V uvodnem poglavju je opisan uvod in cilji diplomske naloge.

Sledi pregled priporočilnih sistemov v teoriji in prikaz različnih sistemov na praktičnih primerih.

V tretjem poglavju je predstavljena implementacija spletne trgovine. Opis implementacije zajema uporabniške zahteve, načrt izdelave spletne trgovine, uporabljene tehnologije, opis razvoja, funkcionalnost spletne trgovine in postavitev v produkcijsko okolje. Na koncu tega poglavja je opisana varnost spletne trgovine.

V četrtem poglavju spletno trgovino, ki smo jo razvili, nadgradimo z implementacijo priporočilnega sistema. Poglavje je razdeljeno na izdelavo prototipa, opis pridobivanja podatkov o naročilih ter samo implementacijo priporočilnega sistema.

V predzadnjem poglavju analiziramo implementacijo spletne trgovine in priporočilnega sistema. V tem poglavju so opisane težave, ki so se pojavile pri implementaciji, analiza časovne zahtevnosti priporočilnega sistema ter možne izboljšave spletne trgovine.

V šestem, zadnjem poglavju, sklenemo diplomsko delo z zaključkom, kjer strnemo ugotovitve, do katerih smo prišli med razvojem in testiranjem spletne trgovine in priporočilnega sistema.

Poglavje 2

Priporočilni sistemi

V tem poglavju opisujemo, kaj so priporočilni sistemi in kako tržniki uporabljajo priporočilne sisteme za izboljšanje prodaje, na koncu poglavja pa predstavljamo primere priporočilnih sistemov, ki jih uporabljajo velike spletne trgovine.

2.1 Ozadje

Priporočilni sistem (angl. Recommendation system) je vse pogosteje uporabljena tehnika v spletnih trgovinah, ki pomaga potrošniku pri izbiri izdelka [13]. Kar je bilo v preteklosti noviteta, je danes postalo resno poslovno orodje. Priporočilni sistemi uporabljajo znanje o izdelkih, ki so lahko s pomočjo strokovnjakov ustvarjeni ročno ali avtomatsko s tehniko podatkovnega rudarjenja. Strokovnjaki pri podatkovnem rudarjenju vnaprej določijo pravila, nato pa sledi avtomatsko učenje na podlagi obnašanja kupca v spletni trgovini.

Izdelki so lahko predlagani glede na splošno prodajo v trgovini, glede na demografske značilnosti kupca, ali glede na analizo prejšnjih nakupov, ki služi kot podlaga za nadaljnje nakupe [15]. Naštete tehnike so del personalizacije na spletu, saj omogočajo, da se stran prilagodi vsakemu potrošniku posebej.

Prilagoditev je ena od poti do uresničitve idej, ki jih je imel Joe Pine v svoji knjigi *Mass Customization* [11]. Pine bi se verjetno strinjal s trditvijo,

ki jo je izrekel Jeff Bezos, izvršni direktor spletne trgovine Amazon.com™: "Če imam na spletu tri milijone potrošnikov, bi moral imeti tri milijone spletnih trgovin" [14]. Pine in Bezos se strinjata tudi, da prodaja enega samega izdelka ni več primerna ter da bi morala podjetja pri spletni prodaji potrošniku ponuditi več možnosti [11].

2.1.1 Izboljševanje prodaje s pomočjo priporočilnih sistemov

Priporočilni sistemi so hkrati podobni in različni od sistemov za trženje ter sistemov za podporo odločanju pri dobavnih verigah [15]. Sistemi za trženje omogočajo podporo tržniku pri odločitvah, kako oglaševati izdelke kupcu. Ponavadi tržniki s pomočjo segmentacije potrošnikov in kategoriziranjem izdelkov v skupine povežejo segmente potrošnikov s kategorijami izdelkov. Nato preko tržnih kampanj spodbujajo potrošnike k nakupu izdelkov iz kategorije izdelkov, namenjenih tej skupini potrošnikov. V nasprotju s sistemi za trženje priporočilni sistemi neposredno komunicirajo s potrošnikom in mu pomagajo najti izdelke, ki jih bo najverjetneje kupil.

Sistemi za podporo odločanja pri dobavnih verigah pomagajo tržnikom napovedati, koliko izdelkov naj naročijo in katerim trgovinam oziroma skladiščem naj izdelke dostavijo. Taki sistemi uporabljajo analitične tehnologije za napovedovanje prodaje izdelkov, ki bodo omogočali pravočasno in zadostno naročanje izdelkov, ter s tem zagotavljajo, da bodo v prodajalnah vedno na voljo izdelki, ki jih potrošniki potrebujejo. Priporočilni sistemi pri tem delujejo drugače. Osredotočajo se na posameznega potrošnika in odgovarjajo na vprašanje, kateri izdelek želi potrošnik kupiti v tem trenutku.

Priporočilni sistemi izboljšujejo prodajo na naslednje načine:

Pretvorba uporabnikov v kupce:

Obiskovalci spletne trgovine pogosto pregledujejo izdelke na spletni strani brez želje, da bi karkoli kupili. Priporočilni sistemi izpostavijo izdelke in s

tem prepričajo obiskovalca, da priporočeni izdelek tudi kupi.

Izboljšanje navzkrižne (angl. cross-sell) prodaje:

Priporočilni sistemi izboljšujejo navzkrižno prodajo preko priporočil izdelkov, ki bi morda zanimali potrošnika. Če so priporočila dobra, se mora povprečna vrednost nakupa povečati. Primer: spletna trgovina na podlagi izdelkov, ki so že v košarici, priporoči dodatne izdelke, ki so najprimernejši glede na trenutno vsebino košarice. Poleg navzkrižne prodaje lahko potrošnikom s pomočjo natančnih priporočil priporočilni sistemi omogočijo izbiro izdelkov, ki jih drugače ne bi našli.

Gradnja zvestobe:

V svetu, kjer je konkurenčna spletna trgovina samo klik ali dva stran, mora biti zvestoba potrošnika osrednja poslovna strategija [12]. Priporočilni sistemi izboljšujejo zvestobo z ustvarjanjem dodane vrednosti odnosu med spletno trgovino in potrošnikom. Stran investira v učenje o njenih potrošnikih in s pomočjo priporočilnih sistemov spremlja navade potrošnikov z željo, da se približa njihovim potrebam. Potrošniki poplačajo investicijo v priporočilni sistem s tem, da se vračajo v trgovino, ki najbolj ustreza njihovim željam. Na zvestobi se lahko gradi tudi tako, da se ustvari odnos med potrošniki. Potrošniki so bodo raje vračali na stran, kjer so ljudje, s katerimi se radi srečujejo.

Povečanje uporabniškega zadovoljstva:

Dobro načrtovan priporočilni sistem lahko poveča zadovoljstvo potrošnika v spletni trgovini. Potrošnikom se bo zdel priporočilni sistem z natančnimi in učinkovitimi priporočili zanimiv ter s privlačnim grafičnim vmesnikom tudi prijeten za uporabo.

2.2 Primeri uporabe priporočilnih sistemov

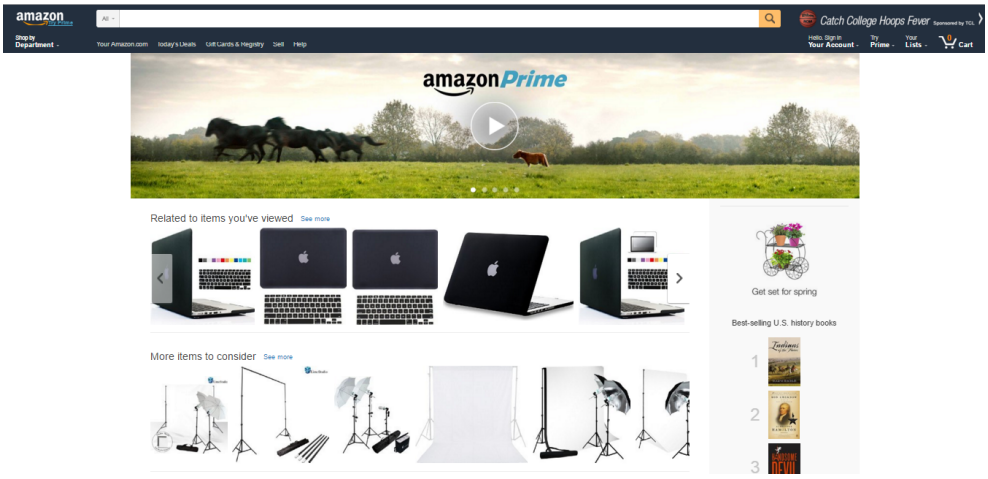
V tem razdelku bomo predstavil tri primere priporočilnih sistemov, ki uporabljajo enega ali več različnih tehnologij priporočilnih sistemov. Za vsako spletno trgovino bomo na kratko predstavili, katere tipe tehnologij priporočilnih sistemov uporablja. Opisane spletne trgovine smo izbrali zaradi njihove popularnosti in števila potrošnikov, ki spletne trgovine uporabljajo. Predstavljene so po abecednem vrstnem redu. Zadnja posodobitev seznama je bila 30. marca 2016.

2.2.1 Amazon.com

Spletno trgovino Amazon.com™ [1] lahko najdemo na spletnem naslovu `http://www.amazon.com`, začetna stran pa je prikazana na sliki 2.1. Prvi priporočilni sistem se na njihovi strani nahaja že takoj pod glavno pasico strani (slika 2.1, zgoraj). S to postavitvijo priporočilnih sistemov nam daje Amazon vedeti, da te sisteme jemljejo zelo resno in da tovrsten način izboljševanja prodaje s pridom izkoriščajo. Pod glavno pasico sta vsebinsko in vizualno postavljena kar dva traka s slikami in napisoma, ki bi v slovenskem prevodu zvenela "Povezano z izdelki, ki ste si jih nazadnje ogledovali"(angl. Related to items you've viewed), na sredini slike 2.1, in "Več izdelkov, ki bi vas morda zanimali"(angl. More items to consider), spodaj na sliki 2.1.

Amazon.com™ zaradi vizualne privlačnosti potrošnikom ne vsiljuje dodatnih besedil, naslovov ali cen izdelkov, temveč potrošnike z elegantnimi slikami privabi, da si s klikom na sliko izdelek podrobneje ogledajo.

V zgoraj naštetih primerih podjetje Amazon.com™ uporablja tehniki navzkrižne prodaje in gradnje zvestobe. Potrošnikom predlaga, da razširijo zanimanje še na druge kategorije izdelkov. Hkrati pa z nevsiljivo ponudbo priporočenih izdelkov potrošniku trgovina daje vedeti, da spletna trgovina sledi njegovim željam in mu želi pri nakupu pomagati.

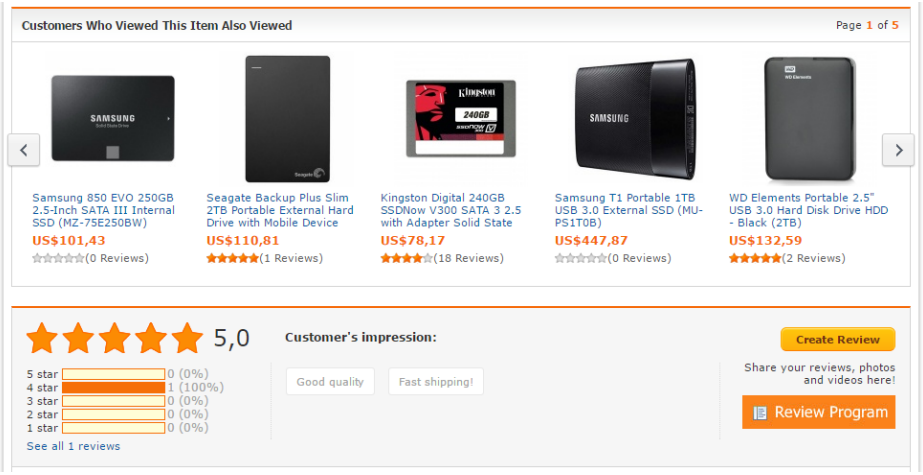


Slika 2.1: Primer priporočilnih sistemov na spletni strani trgovca Amazon.com™

2.2.2 DealExtreme

Spletna trgovina DealExtreme se nahaja na spletnem naslovu `http://www.dx.com` [3]. Podobno kot pri prejšnjem primeru tudi podjetje DealExtreme uporablja priporočilne sisteme že na glavni strani, vendar ti sistemi pridejo do izraza šele ob podrobnem pogledu izdelka. Na koncu opisa izdelkov lahko zasledimo pasico (slika 2.2 zgoraj), ki nagovarja potrošnike, naj si ogledajo izdelke, ki so si jih ogledali drugi potrošniki po tem, ko so zaključili z ogledom trenutnega izdelka. S tem želi podjetje doseči dva učinka. Z velikim naborom izdelkov želijo potrošnika prepričati, da bodo v njihovi spletni trgovini našli izdelek, ki ga iščejo. Prav tako pa z dodatno ponudbo obiskovalca zadržijo na strani, da si ogleda še druge izdelke v njihovi ponudbi.

Drug primer priporočilnega sistema je sistem, kjer potrošniki ocenjujejo izdelek z ocenami in komentarji (slika 2.2 spodaj). Tak sistem je primeren, ker gradi na zvestobi med uporabniki in bodoče potrošnike nagovarja k nakupu izdelka.

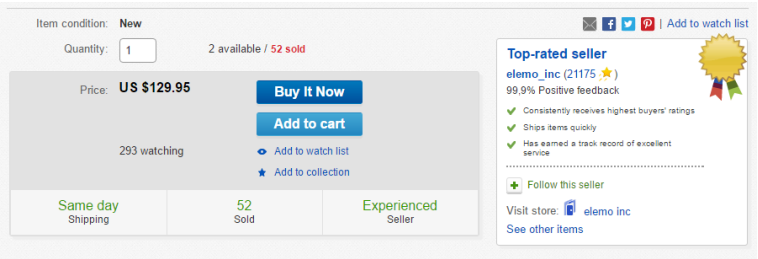


Slika 2.2: Primer dveh priporočilnih sistemov, ki ju uporablja podjetje DealExtreme v svoji spletni trgovini. Prvi (zgornji del slike) predlaga izdelke, ki so si jih drugi uporabniki poleg trenutnega tudi ogledali. Drugi (spodnji del slike) je sistem za oddajanje priporočil o izdelku s pomočjo besedila in ocene od 1 do 5.

2.2.3 eBay

Tretji primer je spletna trgovina eBay™ [4], ki jo lahko najdemo na spletnem naslovu `http://www.ebay.com`. Podjetje eBay™ uporablja sistem z oddajo povratnih informacij (angl. feedback), ki omogoča tako potrošnikom kot prodajalcem dodajanje priporočil o poslovanju s partnerjem. Povratne informacije vsebujejo oceno zadovoljstva (zadovoljen, nevtralno, nezadovoljen) kot tudi besedilni komentar poleg ocene. Povratne informacije v tem primeru služijo kot priporočilni sistem za potrošnike, ki lahko vidijo oceno prodajalca v njegovem osebnem profilu. Povratne informacije se odražajo kot odstotek pozitivne zadovoljnosti (slika 2.3 desno), kjer je ocena 100% najvišja in ocena 0% najmanjša.

Tu podjetje eBay™ uporablja tehniko izboljševanja prodaje, ki smo jo imenovali zvestoba uporabnikov. Bodoči potrošniki se na podlagi ocen prodajalca odločijo za sodelovanje z njim.



Slika 2.3: Podjetje eBay^{TM} uporablja sistem z oddajo povratnih informacij. Povratne informacije se odražajo kot odstotek pozitivne zadovoljnosti.

Poglavje 3

Implementacija spletne trgovine

V tem poglavju bomo opisali celotno implementacijo spletne trgovine. Najprej bomo predstavili določitev uporabniških zahtev. Sledi načrt in arhitektura postavitve spletne trgovine ter predstavitev uporabljenih tehnologij. V nadaljevanju je opisan razvoj spletne trgovine in opis njenih funkcionalnosti. Poglavje sklenemo s prikazom postavitve v produkcijsko okolje in opisom varnosti spletne trgovine.

3.1 Uporabniške zahteve

Namen ugotavljanja uporabniških zahtev je določitev, kaj naročnik želi od produkta, oziroma kaj je produkt zmožen narediti. Pri implementaciji spletne trgovine smo s pomočjo ugotovljenih uporabniških zahtev definirali, kaj se od spletne trgovine pričakuje in kakšne funkcionalnosti naj vsebuje.

Uporabniške zahteve smo razdelili v tri skupine. V prvo skupino sodijo akcije, ki jih lahko izvaja uporabnik, torej obiskovalci in stranke, ki bodo obiskovali spletno trgovino. Obiskovalci so vsi uporabniki, ki uporabljajo spletno trgovino, stranke pa so tisti obiskovalci, ki se odločijo za nakup. V drugo skupino sodijo zahteve oziroma akcije, ki jih lahko izvajajo upravitelji

spletne trgovine. V tretjo skupino sodijo zahteve, ki so namenjene splošni uporabnosti strani.

Uporabniki lahko v spletni trgovini:

- izvedejo registracijo oziroma prijavo ali oddajo zahtevek za ponastavitev gesla,
- urejajo svoj profil, pregledujejo zgodovino naročil, obvestila,
- brskajo po spletni trgovini, uporabljajo filtre za lažje iskanje,
- dodajajo izdelke v košarico in imajo možnost oddaje naročila preko blagajne.

Upravitelji trgovine lahko na strani izvajajo sledeče akcije:

- pregledovanje naročil v določenem časovnem obdobju,
- urejanje kataloga izdelkov in informacij o izdelkih,
- pregled strank,
- upravljanje poročil za določeno časovno obdobje,
- upravljanje naročil (premikanje stanj, odobritve, preklic),
- urejanje statičnih strani,
- kreiranje promocij.

Splošne lastnostni spletne trgovine:

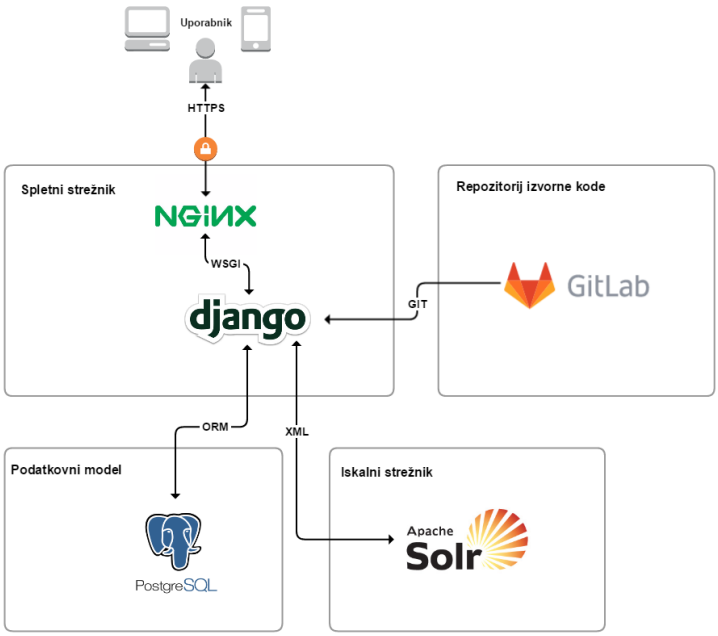
- enostavna za uporabo in prijazna uporabniku,
- mora delovati v vseh modernih brskalnikih,
- čim bolj prilagojena za mobilne in tablične naprave,
- mora biti varna za stranke in upravitelje.

3.2 Načrt izdelave spletne trgovine

V naslednjem koraku smo izdelali načrt spletne trgovine. V okviru tega je bilo potrebno definirati arhitekturo spletne trgovine in podatkovni model, ki sta predstavljena v tem podpoglavju.

3.2.1 Arhitektura spletne trgovine

Arhitektura spletne trgovine je razdeljena na štiri dele. Ko uporabnik zahteva vsebino strani na spletni trgovini, se pošlje zahtevek HTTP preko protokola SSL/TLS strežniku Nginx, ki deluje kot obrnjen posrednik (ang. reverse proxy). Nginx preko vmesnika WSGI posreduje zahtevek aplikaciji Django (slika 3.1 zgoraj). Aplikacija zahtevek sprejme v svojih kontrolerjih in delo razdeli poslovni logiki. Če se izkaže potreba po podatkih iz podatkovne baze, jih ogrodje Django zahteva preko modula za objektno-relacijsko mapiranje (slika 3.1 spodaj levo). V kolikor je uporabniški zahtevek iskalni, aplikacija Django zgenerira nov zahtevek v formatu XML in ga pošlje strežniku Apache Solr, ki aplikaciji odgovori na dano iskanje (slika 3.1 spodaj desno). Ko se izvajanje poslovne logike konča, pošlje ogrodje odgovor nazaj strežniku Nginx v obliki, kot jo je zahteval uporabnik. Strežnik Nginx zaključi komunikacijo, ko odgovori na uporabniško zahtevo HTTP.



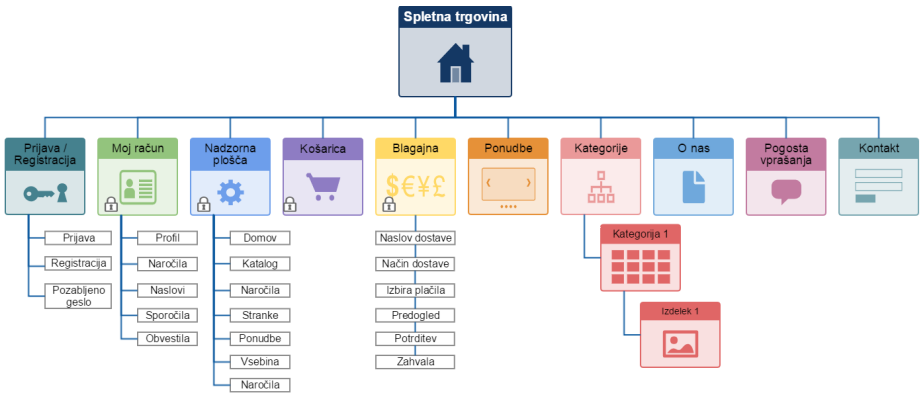
Slika 3.1: Arhitektura spletne trgovine.

3.2.2 Zemljevid in struktura spletišča

Trgovina ima hierarhično spletno strukturo. Razdeljena je na več podkategorij, ki imajo lahko še svoje podkategorije. Tako imajo strani *Prijava / Registracija*, *Moj račun*, *Blagajna* in *Nadzorna plošča* še svoje podstrani, ki so vse, razen podkategorije *Blagajna*, urejene hierarhično. Podkategorija *Blagajna* sledi linearni spletni strukturi, saj si podstrani na blagajni sledijo zaporedno v obliki čarovnika.

Spletna trgovina se poskuša držati principov dobre strukturiranosti spletnega mesta; principa sedmih povezav in treh klikov skrbita, da podstrani v hierarhični ureditvi ni preveč in da lahko uporabnik spletišča pride do večine strani v največ treh klikih.

Na sliki 3.2 je prikazana organizacija spletne trgovine.



Slika 3.2: Zemljevid spletne trgovine.

3.2.3 Načrt podatkovnega modela

Zaradi kompleksnosti podatkovnega modela in preglednosti skice so bile iz načrta izpuščene nekatere tabele, tako da so ostale samo pomembnejše, ki so prikazane na sliki 3.3, ter povezave med njimi.

Partner tabela predstavlja dobavitelje izdelkov spletne trgovine. V tabeli dobaviteljev so shranjeni podatki o zalogi izdelkov ter njihovih cenah.

Catalog tabela služi predstavitvi katalogov. Katalog je nadpomenka za kategorije, izdelke, vrste izdelkov, atribute izdelkov in priporočila izdelkov. Tabela povezuje vse naštete podpomenke med seboj in med dobavitelji.

Order tabela predstavlja naročila. Naročilo vsebuje polja, kot so informacije o stranki, naslov dostave ter način plačila. Naročilo je razdeljeno na posamezno postavko naročila, ki je povezana z izdelkom spletne trgovine in dobaviteljem.

User tabela je namenjena shranjevanju uporabnikov spletne trgovine. Tabela hrani osnovne podatke o uporabnikih, njihovih naročilih in pravicah, imenik naslovov ter informacije o uporabniški seji.

Na sliki 3.3 je prikazan načrt podatkovnega modela spletne trgovine.

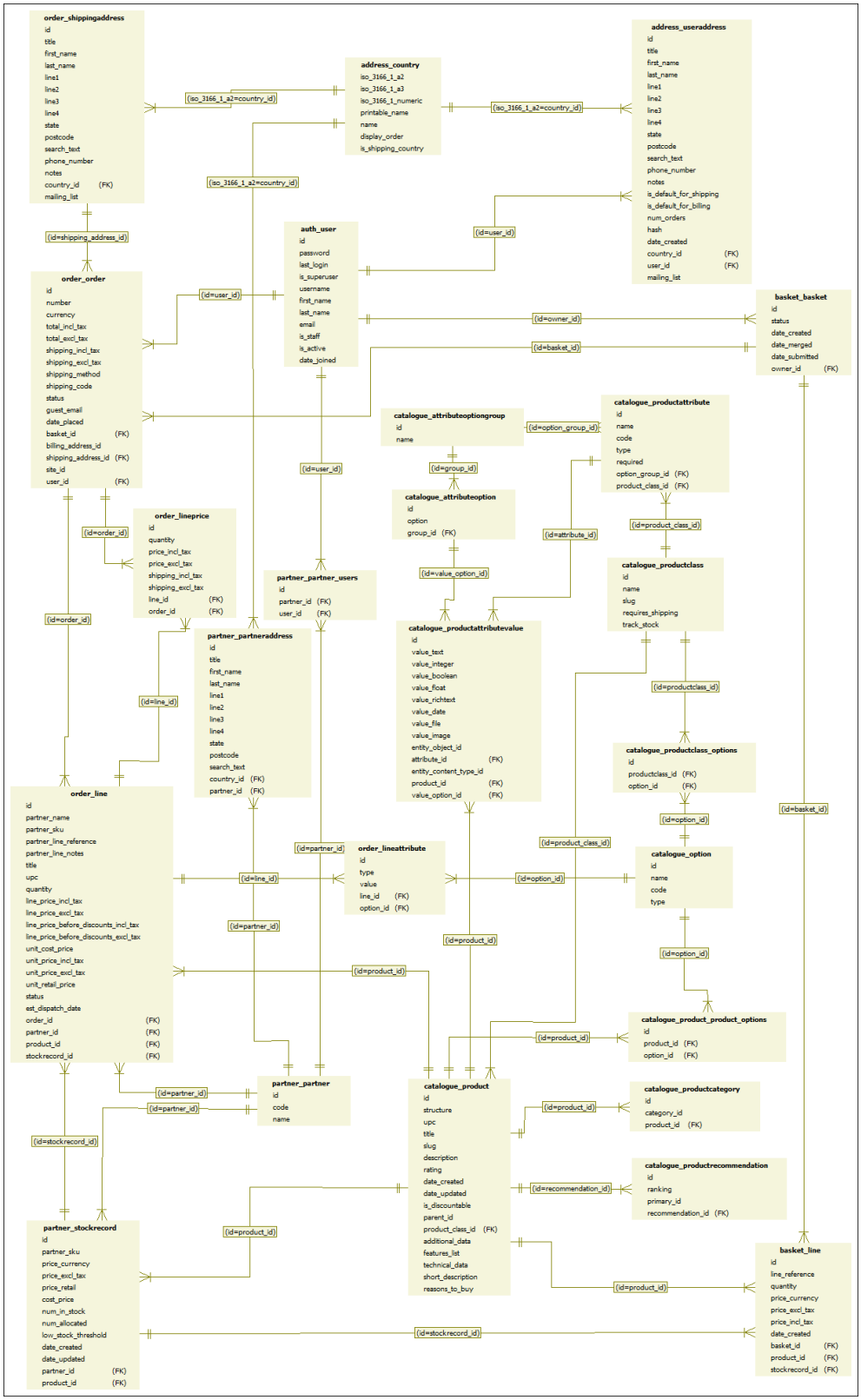
3.3 Uporabljene tehnologije

V tem razdelku predstavljamo tehnologije, ki smo jih uporabili pri implementaciji spletne trgovine in priporočilnega sistema.

3.3.1 Jezik HTML

Jezik HTML (angl. Hypertext Markup Language) je označevalni jezik za opis strukture spletnih strani. Ta označevalni jezik je temeljna tehnologija za gradnjo spletnih strani. Spletni brskalniki preberejo datoteke HTML in jih sprocesirajo v vidno/slušne elemente [7]. Jezik je bil s strani organizacije IETF(Internet Engineering Task Force) uradno definiran leta 1993, zadnja različica (verzije 5) pa je izšla 28. oktobra 2014.

HTML smo uporabljali za določanje elementov, ki smo jih uporabljali na spletni strani, oziroma za določanje strukture spletne strani.



Slika 3.3: Arhitektura podatkovnega modela.

3.3.2 Kaskadne stilske predloge

Kaskadne stilske predloge (ang. Cascading Style Sheets, CSS) so jezik, ki opisuje predstavitev dokumenta, napisanega v jeziku HTML ali drugih označevalnih jezikih, na primer SVG (angl. Scalable Vector Graphic). CSS opisuje, kako naj izgledajo strukturirani elementi na zaslonu, papirju, pri izgovorjavi ali v drugih medijih. Stilske predloge so bile uporabljene v implementirani spletni trgovini za postavitev in oblikovanje elementov.

3.3.3 Jezik Javascript

Jezik Javascript (imenovan tudi ECMAScript) je dinamično tipiziran, interpretiran, višji programski jezik, ki je bil standardiziran leta 1996. Poleg jezika HTML in CSS je dandanes Javascript ena izmed ključnih tehnologij pri gradnji spletnih strani. Zadnja verzija jezika je ECMAScript 7. Uporablja se za komunikacijo z vsebino dokumenta (DOM) ter za ustvarjanje dinamičnosti spletne strani (kot so animacije, definicije dogodkov, spreminjanje stilov elementov, itd.). Pri izdelavi spletne trgovine smo jezik Javascript uporabili za izboljšanje uporabniške izkušnje in povečanje odzivnosti spletne trgovine na manjših napravah.

3.3.4 Jezik Python

Jezik Python je višji programski jezik. Je dinamični jezik, namenjen za splošno uporabo. Pythonova filozofija oblikovanja kode temelji na berljivosti kode in uporablja sintakso, ki programerjem omogoča izražanje konceptov v nekaj vrsticah kode. Python se je prvič pojavil leta 1991, zadnja stabilna verzija pa je bila izdana julija 2016. Pri razvoju spletne trgovine smo jezik Python uporabili za implementacijo zalednega sistema. Prav tako smo s pomočjo jezika Python razvili priporočilni sistem.

3.3.5 Django

Django je odprtokodno spletno ogrodje, spisano v jeziku Python. Ogrodje sledi arhitekturi MVC (angl. Model - View - Controller). Vzdržuje ga Django Software Foundation, ki je neodvisna, neprofitna organizacija. Primarni cilj Djanga je poenostaviti kreiranje kompleksnih, podatkovno gnanih spletnih strani. Sloni na principih ponovne uporabe kode, vtičnikov, hitrega razvoja in principu DRY. Ogrodje med drugim vsebuje tudi modul za objektno-relacijsko mapiranje (ORM), ki posreduje med podatkovnim modelom in relacijsko podatkovno bazo.

Ogrodje Django je pri izdelavi spletne trgovine uporabljeno kot osnova, na kateri je zgrajena spletna trgovina. Preko svojega modela MVC prikazuje komponente trgovine, skrbi za izvajanje zaledne logike in skrbi za upravljanje s podatki v podatkovni bazi.

3.3.6 Oscar

Oscar je ogrodje spletne trgovine v ogrodju Django, namenjeno gradnji domensko vodenih spletišč [10]. Ogrodje je strukturirano tako, da je možno vsak del jedrne funkcionalnosti prilagoditi lastnim potrebam. Modularno jedro tako omogoča pokrivanje širokega spektra zahtev spletnih trgovin. Podjetje Tanget Snowball je z razvojem ogrodja začelo v začetku leta 2012 [9].

Ogrodje Oscar je bilo pri implementaciji spletne trgovine uporabljeno kot osnova za izgradnjo spletne trgovine. Jedrni deli ogrodja so bili dedovani in razširjeni za potrebe implementacije uporabniških zahtev.

3.3.7 Pip

Pip je sistem za upravljanje paketov, ki skrbi za namestitve in vzdrževanje programskih paketov napisanih v jeziku Python. V Python Package Index (PyPI) se lahko najde veliko paketov. Pip je rekurzivni akronim, v slovenščino ga lahko prevedemo kot: Pip namešča pakete (angl. Pip Installs Packages). V verzijah jezika Python 2.7.9 in Python 3.4 ter vseh naslednjih

verzijah je upravljalnik paketov Pip privzeto vključen pri namestitvi poleg Pythona.

V listingih 3.1, 3.2, 3.3, 3.4, 3.5 in 3.6 so predstavljeni glavni paketi, ki so uporabljeni pri implementaciji spletne trgovine.

Listing 3.1: Paketi, uporabljeni za jedro aplikacije.

```
Django==1.8.9                   # django core
Pygments==2.1.1                 # syntax highlighting
django-appconf==1.0.1           # handle config defaults
django-compressor==2.0          # compress for assets
rcssmin==1.0.6                  # CSS Minifier
django-debug-toolbar==1.4       # django debug toolbar
django-extensions==1.6.1        # django extensions
django-extra-views==0.6.4       # class based generic views
decorator==4.0.9                # python decorators
django-treebeard==4.0           # efficient tree impl.
apipkg==1.4                     # namespace and lazy-import
Unidecode==0.04.19              # unicode decode
django-widget-tweaks==1.3       # widget tweaker
Jinja2==2.8                     # general purpose templating
purl==1.2                       # easy URL-building
simplegeneric==0.8.1            # simple generic functions
six==1.10.0                     # django compatibility library
```

Listing 3.2: Paketi, namenjeni za dokumentacijo in prevajanje spletne trgovine.

```
Babel==1.3                      # tools for i18n
Sphinx==1.3.3                   # easy docs tool
sphinx-rtd-theme==0.1.9         # sphinx theme
sphinxcontrib-napoleon==0.4.3 # sphinx "napoleon" extension
alabaster==0.7.7                # theme for Spinx docs
docopt==0.6.2                   # docs options
docutils==0.12                  # docs utils
python-gettext==3.0             # compiling library for i18n
pytz==2015.7                    # timezone definitions
```

Listing 3.3: Paketi, potrebni za delovanje ogrodja Oscar.

```
django-oscar==1.1.1          # django oscar
phonenumbers==6.3.0          # library for phone numbers
pycountry==1.8               # library for countries
Pillow==2.7.0                # image library
```

Listing 3.4: Paketi, namenjeni za iskanje in filtriranje izdelkov.

```
pysolr==3.5.0                # SOLR plugin
MarkupSafe==0.23             # safe XML/HTML for String
Whoosh==2.6.0                # full-text indexing
django-haystack==2.3.2       # search for django
sorl-thumbnail==12.2         # thumbnails for SOLR
```

Listing 3.5: Paketi, uporabljeni pri testiranju spletne aplikacije.

```
WebTest==2.0.17              # helper for WSGI test
pytest==2.8.5                # testing for python
pytest-cache==1.0            # caching across test runs
pytest-cov==2.2.0            # measuring coverage
pytest-django==2.9.1         # tests for django
pytest-xdist==1.13.1         # for distributed testing
coverage==3.7.1              # code coverage measurement
coveralls==0.4.4             # show coverage stats online
beautifulsoup4==4.4.1        # screen-scraping library
django-webtest==1.7.8        # web tests
django-tables2==1.0.7        # creating HTML tables
flake8==2.5.1                # modular source code checker
execnet==1.4.1               # rapid multi-Python deployment
mock==1.0.1                  # rolling backport of unittest
nose==1.3.7                  # nose extends unittest
spec==0.11.1                 # specification output for nose
path.py==8.1.2               # module wrapper for os.path
pep8==1.7.0                  # style guide checker
pyflakes==1.0.0              # passive checker
tox==1.8.1                   # virtualenv-based automation
```

Listing 3.6: Paketi, namenjeni vzpostavitvi okolja za spletno trgovino.

```
WebOb==1.5.1                    # WSGI request/response object
Werkzeug==0.9.6                 # tools for web dev
colorama==0.3.6                 # colored terminal text
psycopg2==2.6.1                 # PostgreSQL driver
py==1.4.31                      # library for cross-python path
whitenoise==2.0.6               # static file serving for WSGI
waitress==0.8.10                # waitress WSGI server
virtualenv==14.0.6              # virtual environment builder
```

3.3.8 Podatkovna baza PostgreSQL

Podatkovna baza PostgreSQL, pogosto imenovana Postgres, je objektna relacijska podatkovna baza. Kot strežnik podatkovne baze je njena glavna funkcija varno hranjenje podatkov in odgovarjanje na zahteve aplikacij. PostgreSQL je prosto dostopna in odprtokodna programska oprema, ki je izdana pod PostgreSQL licenco.

Razvita spletna trgovina za upravljanje s podatki sloni na podatkovni bazi PostgreSQL. Aplikacija s programskim ogrodjem Django preko modula za objektno-relacijsko mapiranje (ORM) mapira modele (Pythonove objekte) v PostgreSQL podatkovno bazo. Mapiranje poteka tako, da je model (Pythonov objekt) preslikan kot vrstica v tabeli v podatkovni bazi. Na ta način lahko aplikacija z uporabo ORM-ja enostavno pridobiva in shranjuje podatke.

3.3.9 Apache Solr

Apache Solr je odprtokodna platforma za iskanje, spisana v jeziku Java. Podpira iskanje po celotnem besedilu, poudarjanje zadetkov, plastno iskanje (angl. faceted search), indeksiranje v realnem času ter dinamično povezovanje in integracijo s podatkovno bazo. Aplikacija Apache Solr deluje kot samostojen iskalni strežnik, pri tem pa uporablja javansko iskalno knjižnico Apache Lucene, ki je jedro iskanja po celotnem besedilu. Preko svojega vmesnika

HTTP/XML in JSON API omogoča dostop večini priljubljenih programskih jezikov. Platforma Apache Solr uporablja zunanje konfiguracije za potrebe prilagoditve različnim aplikacijam, ne da bi posegala v njeno izvorno kodo.

Apache Solr spletni trgovini zagotavlja mehanizem za iskanje po katalogu izdelkov, ki jih spletna trgovina ponuja v svojem naboru. Iskalni mehanizem vključuje iskanje preko iskalne vrstice v glavi strani ter filtriranje izdelkov v sekciji kategorij.

3.3.10 GitLab

Aplikacija GitLab je orodje, ki ponuja repozitorij Git za gostovanje in verzioniranje kode. Poleg verzioniranja izvorne kode aplikacija omogoča tudi pregledovanje kode, sledenje hroščev in uporabniških zahtev, pisanje dokumentacije in avtomatsko gradnjo projekta. GitLab je projekt skupnosti, v katerem je sodelovalo preko 1000 ljudi s celotnega sveta.

V procesu izdelave spletne trgovine je bila aplikacija GitLab uporabljena kot repozitorij kode ter za sledenje uporabniških zahtev. Med gradnjo trgovine smo projekt sproti dokumentirali. Po realizaciji uporabniških zahtev smo s pomočjo aplikacije GitLab sledili vsem hroščem, ki so se pojavili pri razvoju.

3.4 Opis razvoja

V tem poglavju je opisan sam postopek implementacije spletne trgovine ter težave, ki so se pri razvoju (oziroma implementaciji) pojavile.

3.4.1 Postavitev Git Lab projekta in definiranje mejnikov projekta

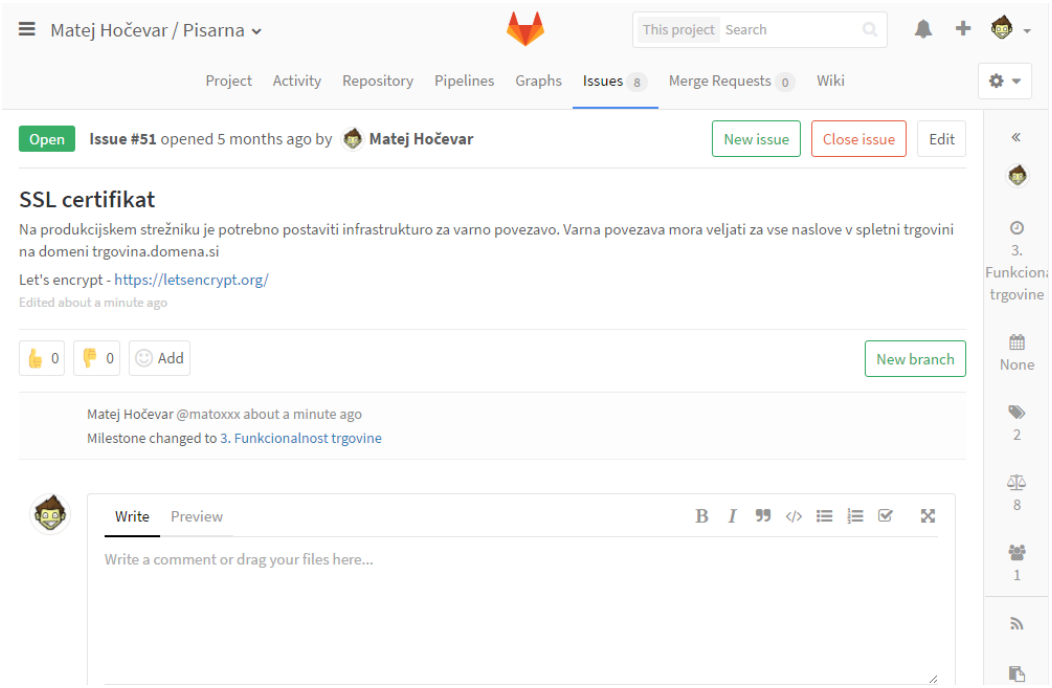
Razvoj projekta smo začeli s postavitvijo računa v spletni aplikaciji GitLab. Poleg repozitorija kode aplikacija GitLab ponuja tudi sistem za prijavljanje in sledenje uporabniških zahtev ter nalog ter pisanje projektne dokumentacije.

Mejnike projekta smo razdelili na pet delov, ki so si sledili linearno, ter vsakemu mejniku namenili 14 dnevni rok.

Mejniki projekta:

1. postavitev okolja,
2. postavitev podatkovnega modela in vnos podatkov,
3. izdelava odjemalčevega dela spletne trgovine,
4. predelava funkcionalnosti spletne trgovine,
5. odprava hroščev in napak.

Na sliki 3.4 je primer prikaza ene izmed nalog v aplikaciji GitLab.



Slika 3.4: Primer izgleda aplikacije GitLab

3.4.2 Oblikovanje izgleda spletne trgovine

Ob namestitvi vtičnika Oscar dobimo osnovni izgled spletne trgovine. Izgled smo s pomočjo kaskadnih stilskih predlog (angl. Cascading Style Sheets, CSS) preoblikovali v želeno rumeno, modro in belo barvno kombinacijo. Deli spletne trgovine so bili preurejeni, da bolje sovpadajo s podobo celotne spletne trgovine. Trgovina je bila prilagojena tudi za naprave z manjšimi zasloni. Podrobnejši izgled spletne trgovine je predstavljen v poglavju 3.5.

3.4.3 Postavitev osnovne strukture spletne trgovine

Projekt spletne trgovine je razdeljen na več poddirektorijev, ki so opisani spodaj, drevesna struktura pa je še dodatno predstavljena v listingu 3.7:

- ***locale*:** V tem direktoriju se nahajajo prevodi kataloga sporočil.
- ***office*:** Glavni direktorij, ki vsebuje vse module spletne trgovine, ločene na poddirektorije, glede na njegovo funkcionalnost. Koda posameznih aplikacij je nadalje ločena v posamezne datoteke, tako da aplikacije sledijo modelu MVC in principu DRY, kot je vidno v drevesni strukturi starša *basket*. V tem direktoriju se poleg aplikacij nahajajo tudi predloge *templates/*, ki jih moduli uporabljajo za odgovor na uporabniške zahteve.
- ***officeShop*:** Direktorij z nastavitvami projekta ter spletne trgovine *settings.py* in *settings_local.py*. V direktoriju se poleg glavnega usmerjevalnika *urls.py* nahaja tudi direktorij z medijskim gradivom *public/*, ki je serviran na spletišču.
- ***static*:** Direktorij, namenjen statičnim knjižnicam, ki jih spletna trgovina uporablja za izgled in funkcionalnost spletne trgovine. V poddirektorijih se nahajajo koda CSS in JavaScript, pisave, ikone ter slike.

Listing 3.7: Drevesna struktura projekta spletne trgovine.

```
Office webshop project
|----locale
|----office
    |----address
    |----basket
         |----abstract_models.py
         |----admin.py
         |----app.py
         |----forms.py
         |----models.py
         |----utils.py
         |----views.py
    |----catalogue
    |----checkout
    |----customer
    |----dashboard
    |----fixtures
    |----migrations
    |----offer
    |----order
    |----partner
    |----payment
    |----promotions
    |----search
    |----shipping
    |----templates
|----officeShop
    |----public
    |----settings.py
    |----settings_local.py
    |----urls.py
|----static
```

3.4.4 Prevod kataloga sporočil v slovenščino

Ogrodje Django ponuja preprost način za prevod vseh sporočil, ki se pojavljajo pri uporabi spletne trgovine. V procesu internacionalizacije spletne trgovine je potrebno posebno označevati dele kode, ki jih želimo imeti prevedene oziroma prilagojene jezikom ter kulturam. Ogrodje poskrbi, da se vse označbe prevodov zberejo na enem mestu, kjer jih prevajalci prevedejo. Primer prevoda dela kataloga je podan v listingu 3.8, medtem ko je v listingu 3.9 prikazan ukaz, ki zgradi sporočilni katalog.

Listing 3.8: Primer prevoda v katalogu sporočil v datoteki django.po.

```
msgid "%(count)d notification archived"
msgid_plural "%(count)d notifications archived"
msgstr[0] "%(count)d obvestilo arhiviranih"
msgstr[1] "%(count)d obvestila arhivirana"
msgstr[2] "%(count)d obvestila arhivirana"
msgstr[3] "%(count)d obvestil arhiviranih"
```

Listing 3.9: Ukazi za izgradnjo sporočilnega kataloga.

```
    python manage.py makemessages --symlinks --locale=sl -e .
   html,.template,.py,.txt --no-location
```

Z zgornjim ukazom ogrodje Django prenese vse označbe sporočil iz predlog v katalog sporočil v datoteko *django.po*, ki se nahaja v direktoriju *locale/sl/LC_MESSAGES*.

Ko je datoteka *django.po* prevedena, se jo z ukazom, ki je prikazan v listingu 3.10, prevede v binarno datoteko *django.mo*, v kateri so shranjeni vsi prevodi za dan jezik in jih ogrodje uporabi pri pošiljanju predlog uporabniku.

Listing 3.10: Ukazi za prevod sporočilnega kataloga v binarno datoteko *django.mo*.

```
    python manage.py compilemessages
```

3.4.5 Uvoz kataloga izdelkov

Poleg ročnega vnosa izdelkov ponuja spletna trgovina tudi vnos večih izdelkov hkrati. Izdelki morajo biti vpisani v datoteki formata CSV z natančno določenim zaporedjem stolpcev, da jih sistem prepozna in pravilno shrani v podatkovno bazo. Spletna trgovina ponuja tri načine uvoza izdelkov, ki so predstavljeni spodaj.

V primeru, da upravitelj strani želi v spletno trgovino uvoziti več izdelkov hkrati, mora oblikovati datoteko v formatu CSV, ki jo nato preko ukaza, prikazanega v listingu 3.11, uvozi v spletno trgovino.

Listing 3.11: Ukaz za uvoz izdelkov preko datoteke *store_products.csv*.

```
python manage.py oscar_import_catalogue pisarna/fixtures/
   store_products.csv --delimiter=";" --flush
```

Po vnosu izdelkov ima upravitelj možnost uvoza atributov izdelka, ki se jih uporablja pri filtriranju izdelkov v katalogu, z ukazom, prikazanim v listingu 3.12.

Listing 3.12: Uvoz atributov v spletno trgovino.

```
python manage.py oscar_import_catalogue_attributes pisarna/
   fixtures/store_product_attrs.csv --delimiter=";" --flush
```

Za konec lahko upravitelj z ukazom, prikazanim v listingu 3.13, preko terminala uvozi slike izdelkov, kjer imena slike kažejo na kataloško številko izdelka.

Listing 3.13: Ukaz za uvoz slik v spletno trgovino.

```
python manage.py oscar_import_catalogue_images  pisarna/
   fixtures/store_images.zip
```

3.4.6 Implementacija dinamičnega iskanja

Vtičnik Django Oscar omogoča dinamično iskanje izdelkov s pomočjo iskalnega strežnika Apache Solr [9].

V procesu implementacije je bilo dodanih več kot deset polj, ki se jih uporablja pri iskanju. Za prikaz ideje implementacije bosta tukaj predstavljeni dve polji. Polja, ki jih želimo uporabljati pri iskanju, je potrebno dodati v proces dinamičnega iskanja. Za začetek smo v datoteko search_indexes.py v razredu *ProductIndex* definirali nova polja:

```
class ProductIndex(indexes.SearchIndex, indexes.Indexable):
    # ...
    color_print = indexes.BooleanField(null=True, faceted=
        True)
    speed_print = indexes.IntegerField(null=True, faceted=
        True)
```

V isti datoteki (nižje) je potrebno implementirati metode, ki pripravijo vrednosti za iskanje posameznega atributa izdelka v pravem formatu:

```
    def prepare_color_print(self, obj):
        try:
            return obj.attribute_values.get(attribute__code='
                color_print').value_boolean
        except ProductAttributeValue.DoesNotExist:
            return None

    def prepare_speed_print(self, obj):
        try:
            return int(obj.attribute_values.get(
                attribute__code='speed_print').value_integer)
        except ProductAttributeValue.DoesNotExist:
            return None
```

Proces iskanja poteka tako, da odjemalec izvede akcijo iskanja ali filtriranja. Zahtevek se pošlje strežniku spletne trgovine, ki prepozna, da gre za zahtevek iskanja po izdelkih. Da je zahtevek uspešno prepoznan, je potrebno v konfiguracijski datoteki *settings.py* definirati atribute, ki se uporabljajo pri iskanju oziroma filtriranju izdelkov:

```
OSCAR_SEARCH_FACETS = {
    'fields': OrderedDict([
```

```
        ('color_print', {'name': _('Color printing'), 'field'
            : 'color_print'}),
        ('speed_print', {'name': _('Printing speed'), 'field'
            : 'speed_print'}),
    ]),
    'queries': OrderedDict([
        ('color_print',
        {
             'name': _('Color printing'),
             'field': 'color_print',
             'queries': [
                 (_('Yes'), True),
                 (_('No'), False),
             ]
        }),
        ('speed_print',
        {
             'name': _('Printing speed'),
             'field': 'speed_print',
             'queries': [
                 (_('0 to 20'), u'[0 TO 20]'),
                 (_('20 to 40'), u'[20 TO 40]'),
                 (_('40 to 60'), u'[40 TO 60]'),
                 (_('60+'), u'[60 TO *]'),
             ]
        }),
    ]),
}
```

Strežnik oblikuje nov zahtevek in ga pošlje iskalnemu strežniku Apache Solr skupaj z iskalnimi parametri, ki jih je zahteval odjemalec.

Iskalni strežnik, ki teče v ločenem procesu, posluša na vratih, ki so določena v konfiguraciji, in odgovarja na iskalne zahteve. V konfiguracijski shemi z imenom *schema.xml* je potrebno dodati novo dodana polja, da zna iskalni strežnik prepoznati zahtevane atribute:

```
<field name="barvno" type="boolean" indexed="true" stored="
   true" multiValued="false" />
```

```
<field name="barvno_exact" type="boolean" indexed="true"
   stored="true" multiValued="false" />
<field name="hitrost_izpisa" type="long" indexed="true"
   stored="true" multiValued="false" />
<field name="hitrost_izpisa_exact" type="long" indexed="true"
    stored="true" multiValued="false" />
```

Iskalni strežnik glede na dano iskanje spletni trgovini vrača izdelke v formatu XML.

3.4.7 Odpravljanje težav in hroščev

V tem razdelku bomo opisali nekaj glavnih težav in programskih hroščev, na katere smo naleteli pri implementaciji spletne trgovine.

Določitev davčne stopnje za izdelke

Prva težava, na katero smo naleteli pri implementaciji spletne trgovine, je bila nastavitev slovenske davčne stopnje. Vtičnik Oscar sicer ponuja nastavitev davčne stopnje, vendar je ta nastavitev mogoča samo za vsakega dobavitelja posebej in ni splošna. Prav tako je dokumentacija za nastavitev davčne stopnje v vtičniku Oscar pomanjkljivo dokumentirana, zato se je bilo potrebno poglobiti in razvozlati kodo spletne trgovine za določitev stopnje. Spletno trgovino želimo uporabljati v slovenskem okolju, zato je bilo potrebno davčno stopnjo implementirati globalno za vse izdelke. Listingi 3.14, 3.15 in 3.16 prikazujejo potrebne popravke kode, ki omogoča, da se vsem izdelkom ob nakupu zaračuna 22% davek.

Listing 3.14: Definicija konstante davčne stopnje v konfiguracijski datoteki *settings.py*

```
OSCAR_DEFAULT_TAX = "0.22"
```

Listing 3.15: Strategija *SL*, ki definira slovensko davčno stopnjo.

```
class SL(UseFirstStockRecord, StockRequired, FixedRateTax,
   Structured):
```

```
    # Use SL VAT rate (as of December 2013)
    rate = D(getattr(settings, 'OSCAR_DEFAULT_TAX', "0.22"))
```

Listing 3.16: Definicija, ki določa, da naj splošna strategija za računanje davka uporablja strategijo *SL*.

```
def strategy(self, request=None, user=None, **kwargs):
    """
    Return an instanticated strategy instance
    """
    return SL(request)
```

Nastavitev načina dostave in plačila

Na podobno težavo kot pri računanju davka smo naleteli pri implementaciji načina dostave in plačila. Vtičnik Oscar ne ponuja opcije, pri kateri bi stranka račun za izdelke plačala po oddaji naročila. Težavo smo rešili tako, da smo sami sprogramirali plačilno metodo, ki plačilo izdelka preloži na termin po oddaji naročila. V listingu 3.17 je prikazana implementacija metode, ki vrača vse metode plačila.

Listing 3.17: Razred *Repository*, ki vrača metode plačila za izbranega uporabnika in košarico.

```
class Repository(object):
    """
    Repository class responsible for returning ShippingMethod
    objects for a given user, basket etc
    """
    methods = (shipping_methods.StaffDelivery(),
       shipping_methods.PersonalPickup())
```

Vtičnik prav tako ne ponuja načinov dostave. Implementirali smo dva nova načina dostave (koda je prikazana v listingu 3.18), med katerima lahko stranke izbirajo; gre za osebni prevzem na sedežu podjetja oziroma dostavo na dom, ki ga opravi osebje trgovine.

Listing 3.18: Definicija metod dostave za osebni prevzem in dostave, ki ga opravi osebje trgovine.

```
class StaffDelivery(Base):
    code = 'staff-delivery'
    name = _('Staff delivery')
    description = _('order is going to be delivered by our
       personnel')

    def calculate(self, basket):
        return prices.Price(
            currency=basket.currency,
            excl_tax=D('0.00'), tax=D('0.00'))


class PersonalPickup(Base):
    code = 'personal-pickup'
    name = _('Personal pickup')
    description = _('in our storage house <br> every workday
       between 8h & 15h 0\'clock')

    def calculate(self, basket):
        return prices.Price(
            currency=basket.currency,
            excl_tax=D('0.00'), tax=D('0.00'))
```

Prilagajanje spletišča za manjše zaslone

Problem nastane, ko je potrebno za različne velikosti zaslona drugače oblikovati enake elemente v drevesu spletišča. Za rešitev tega problema smo uporabili medijske poizvedbe v jeziku CSS, ki omogočajo nastavitev stilov za posamezne elemente glede na napravo oziroma velikost zaslona, ki jo uporablja kupec. Primer tovrstne medijske poizvedbe je prikazan v listingu 3.19.

Listing 3.19: Primer uporabe medijskih poizvedb v jeziku CSS.

```
@media (max-width: 480px) {
  .nav-collapse {
```

```
    -webkit-transform: none;
  }
}
```

3.5 Funkcionalnost spletne trgovine

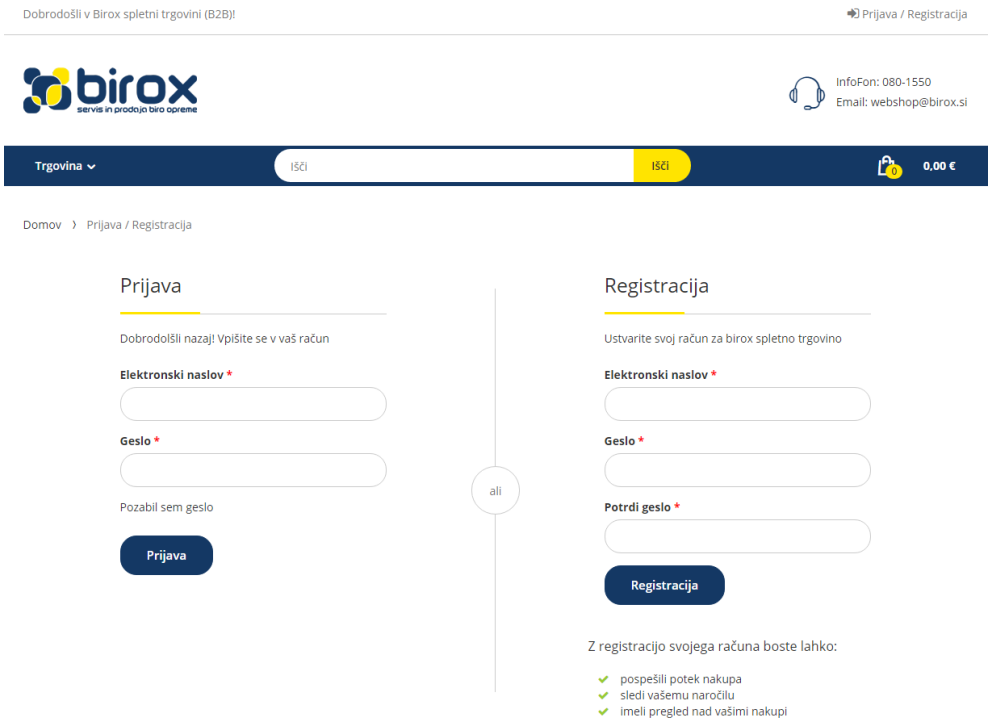
V tem podpoglavju so predstavljene glavne funkcionalnosti spletne trgovine.

3.5.1 Registracija, prijava in ponastavitev gesla

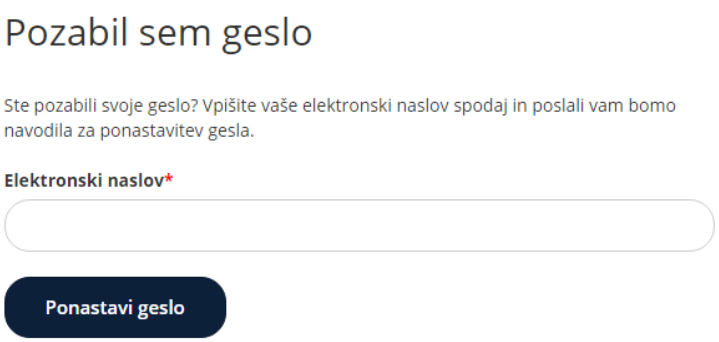
V primeru, da uporabnik ne želi samo pregledovati izdelkov v spletni trgovini, mora ustvariti svoj račun. Svoj račun lahko ustvari na podstrani *Prijava / Registracija*, kjer mora vpisati svoj elektronski naslov in geslo (slika 3.5, desno). Z registracijo novega računa uporabnik pridobi pravico uporabe akcij, ki so opisane v razdelku *Uporabniške akcije*. Uporabnik po registraciji dobi na svoj elektronski naslov sporočilo o uspešni registraciji s povezavo, ki ga vrne v spletno trgovino.

Po uspešni registraciji se uporabnik lahko prijavi v sistem preko podstrani *Prijava / Registracija* (na sliki 3.5 levo). V kolikor sta elektronski naslov in geslo pravilna, je uporabnik preusmerjen na prvo stran trgovine, kjer v glavi strani lahko vidi mogoče akcije za prijavljenega uporabnika kot so *Moj račun*, *Odjava* ter *Nadzorna plošča* (če ima uporabnik pravice upravitelja strani).

V kolikor prijava v spletno trgovino ni uspešna, lahko uporabnik zahteva ponastavitev gesla. Forma za ponastavitev gesla je dostopna s povezavo nad gumbom *Prijava* (povezava "Pozabil sem geslo", na sliki 3.5 na sredini levo). Od uporabnika se zahteva vpis elektronskega naslova, kot je vidno sliki 3.6. Če sistem najde vpisani elektronski naslov v podatkovni bazi, se na ta naslov pošljejo navodila za ponastavitev gesla.



Slika 3.5: Prikaz strani za prijavo oziroma registracijo uporabnika na spletni strani trgovine.



Slika 3.6: Obrazec za pozabljeno geslo.

3.5.2 Uporabniške akcije

Podstran, kjer lahko uporabnik ureja svoje nastavitve (slika 3.7), je dostopna s klikom na povezavo *Moj Račun* v glavi strani. Na podstrani *Moj račun* lahko uporabnik izvaja akcije, kot so menjava gesla, urejanje osnovnih podatkov in izbris računa. Zgodovina opravljenih naročil je vidna s klikom na zavihek *Zgodovina naročil*, kjer ima uporabnik tudi možnost filtriranja svojih naročil. Ker lahko uporabnik uporablja več naslovov za dostavo, lahko te naslove ureja v zavihku *Imenik naslovov*. V zavihku *Zgodovina poslanih sporočil* ima uporabnik pregled nad vsemi sporočili, ki so bila zanj ustvarjena in poslana v njegov predal elektronske pošte. Zadnja povezava v meniju so *Obvestila*. V tem zavihku uporabnik dobiva obvestila o zalogi izdelka, za katerega je izrazil zanimanje.

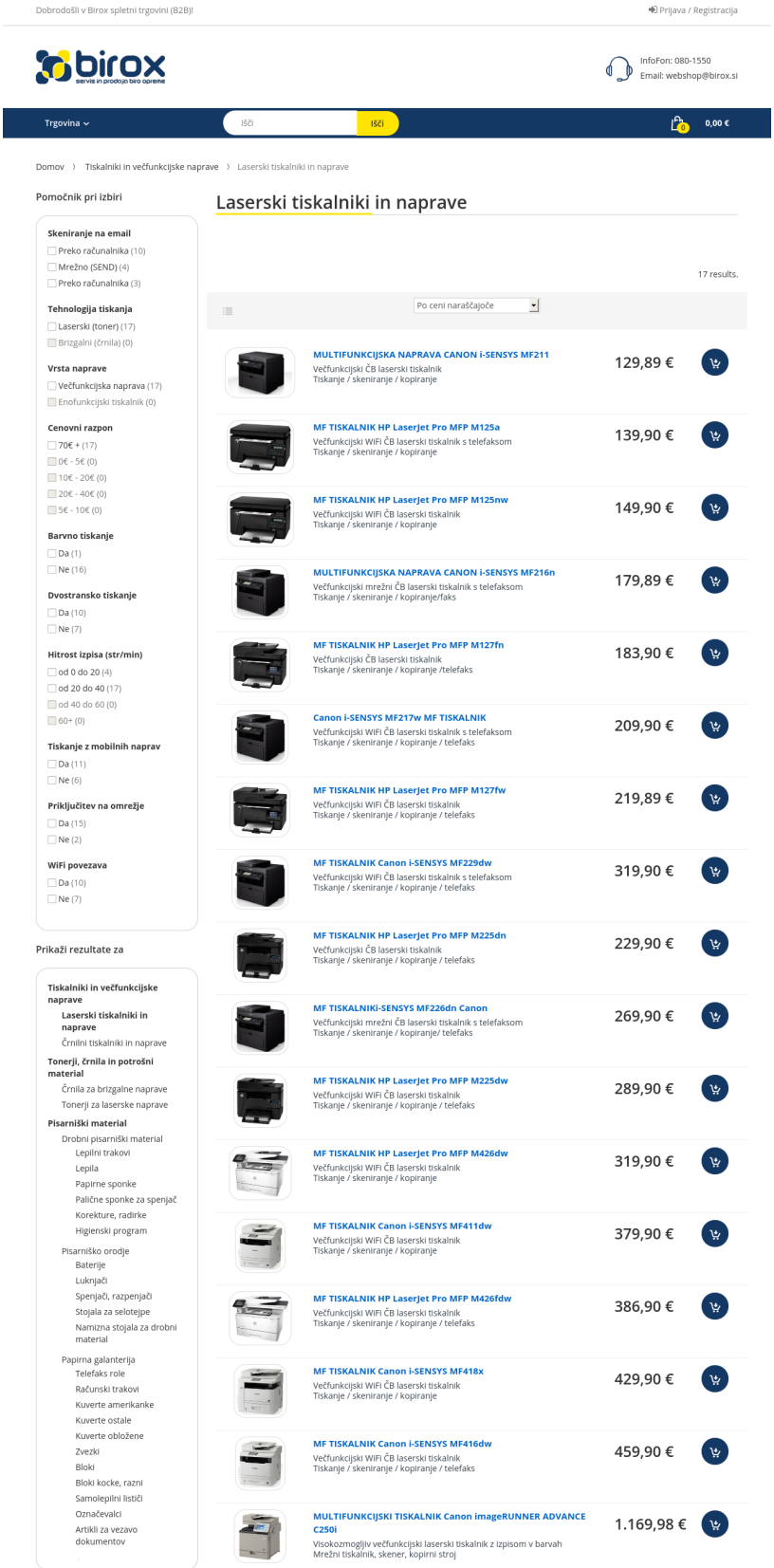
Slika 3.7: Podstran za pregled in urejanje uporabniških nastavitev.

3.5.3 Pregled kategorij in izdelkov

Dostop do kategorije izdelkov je mogoč preko spustnega menija pod logotipom strani ali preko sličic na glavni strani, ki predstavljajo kategorije. Kot je prikazano na sliki 3.8, ima uporabnik možnost urejanja izdelkov po ceni, priljubljenosti, imenu ali času dodajanja. Urejevalnik je na sliki 3.8 viden v sivem pasu pod naslovom kategorije.

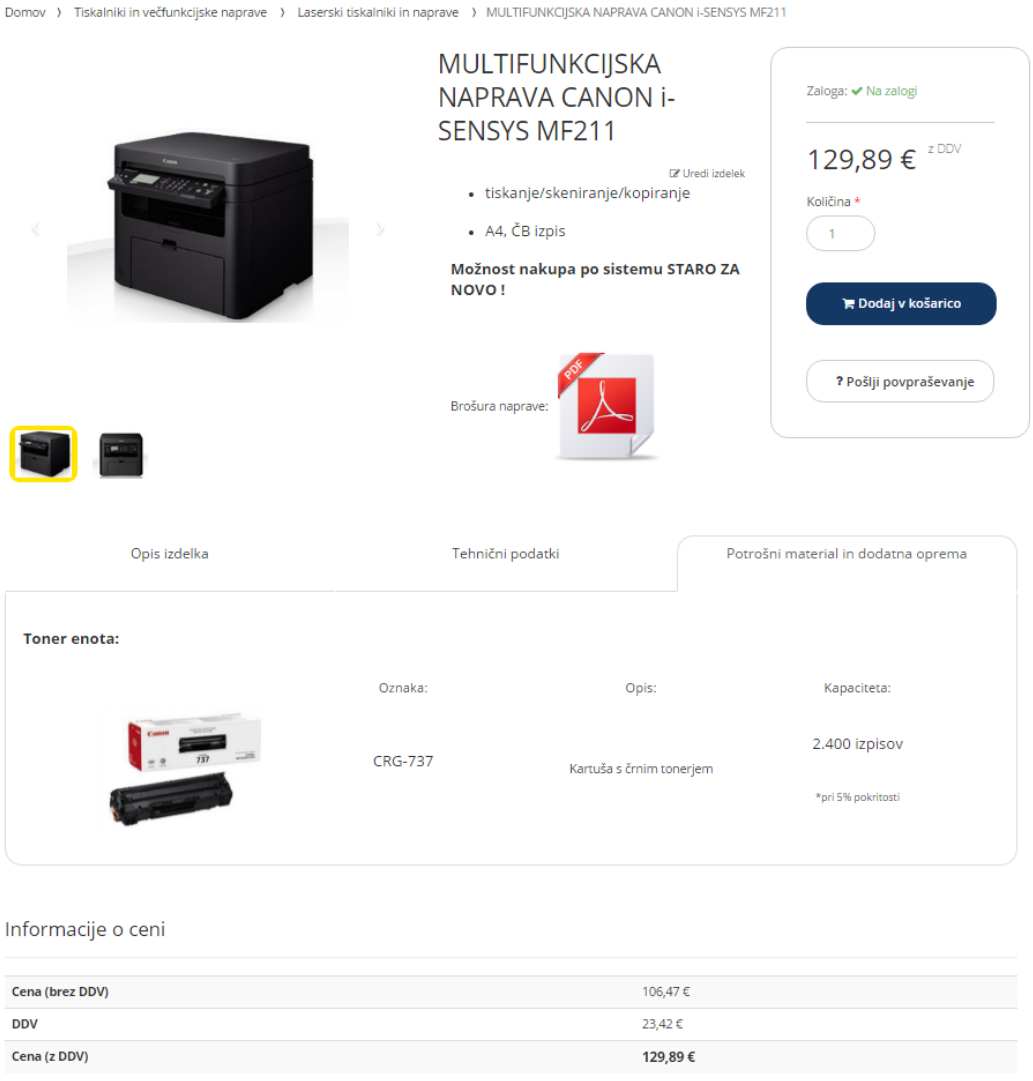
V modrem pasu pod glavo strani na sliki 3.8 se nahaja iskalnik izdelkov v spletni trgovini. Uporabniku iskalnik glede na iskalni niz sproti predlaga izdelke, ki ustrezajo iskalnemu nizu. S klikom na predlog je uporabnik preusmerjen na podrobnejši pregled izdelka.

Poleg osnovnega urejanja izdelkov in iskanja izdelkov je uporabniku na voljo tudi podrobnejše filtriranje prikaza izdelkov (na sliki 3.8 levo). Podrobnejši filter se nahaja v stranski koloni na levi strani. Uporabnik lahko s kliki na željene lastnosti izdelka zoža pogled izdelkov v kategoriji oziroma zmanjša število prikazanih izdelkov.



Slika 3.8: Kategorije z izdelki.

Ko uporabnik najde pravi izdelek, se lahko s klikom nanj premakne v podrobnejši pregled izdelka (slika 3.9). Tu lahko uporabnik podrobneje pregleda opis izdelka, tehnične podatke ter ostale informacije.

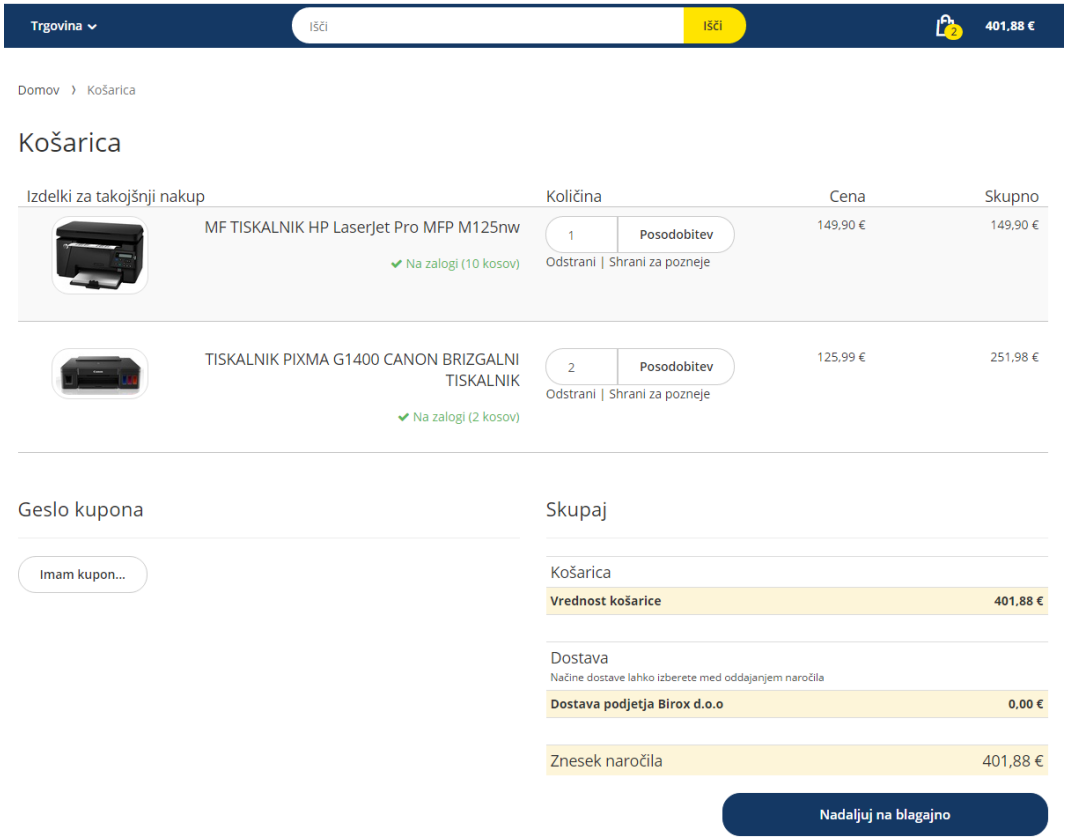


Slika 3.9: Podrobnejši pregled izdelka.

3.5.4 Naročanje izdelkov

Naročanje izdelkov poteka v večih korakih, ki si sledijo v linearni hierarhiji. Uporabnik lahko korake do zaključka naročila spremlja v meniju na vrhu strani.

1. korak: Pregled košarice. Tu ima uporabnik možnost pregleda in ureditve trenutne košarice (slika 3.10). Ko je z vsebino košarice zadovoljen, nadaljuje na blagajno.

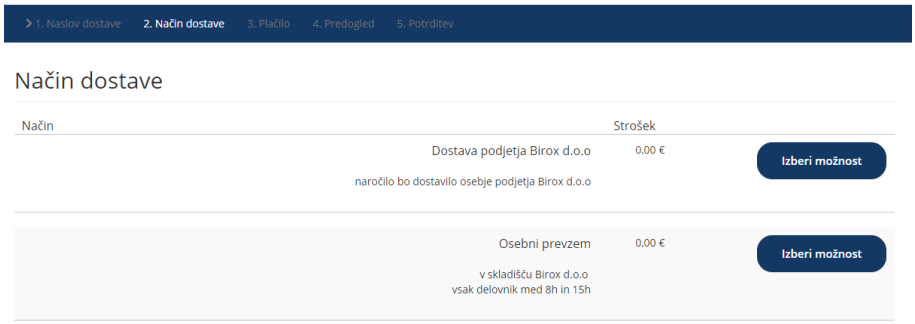


Slika 3.10: Prikaz vsebine uporabnikove košarice.

2. korak: Uporabnik v tem koraku izbere, na katero ime in naslov želi, da se naročilo naslovi. Tu ima uporabnik možnost, da izbere naslov, ki ga

ima v *Imeniku naslovov*, ali da vnese novega.

3. korak: Po izbiri naslova mora uporabnik izbrati način dostave (slika 3.11). Uporabnik izbira med dostavo podjetja in osebnim prevzemom.



Slika 3.11: Izbira načina dostave naročila na blagajni.

4. korak: Sledi izbira plačila. Ker trenutno ni možnosti spletnega plačevanja, je edina možnost plačilo po povzetju.

5. korak: Po izbiri plačila ponovno sledi predogled naročila. Za razliko od 1. koraka se povzamejo še izbira naslova, način dostave in plačilo. V kolikor je uporabnik zadovoljen s predogledom naročila, lahko naročilo odda s pritiskom na gumb *Oddaja naročila*.

6. korak: V zadnjem koraku se uporabniku prikaže zahvala za naročilo ter dodatni napotki v primeru težav. V tem času se uporabniku pošlje tudi elektronsko sporočilo s povzetkom naročila. Primer elektronskega sporočila je prikazan na sliki 3.12.

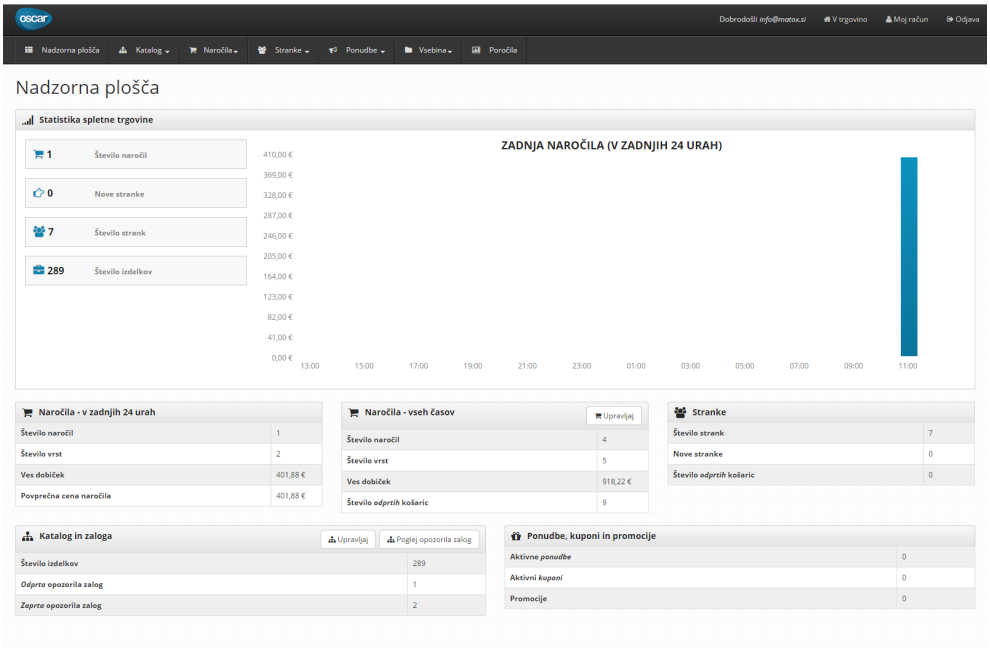
Slika 3.12: Primer elektronskega sporočila po oddanem naročilu strani.

3.5.5 Nadzorna plošča za upravitelja strani

Na prvi strani nadzorne plošče je pregledna plošča, ki povzema statistiko v zadnjih 24 urah (slika 3.13). Statistika vsebuje informacije o zadnjih nakupih, novih uporabnikih, odprtih košaricah, itd.

Na podstraneh nadzorne plošče lahko upravitelj spletne trgovine pregleduje in ureja naročila, ureja podatke o strankah, posodablja izdelke, slike izdelkov in informacije o njih, ali dodaja aktualne ponudbe v spletno trgovino. Prav tako lahko spreminja vsebino statičnih strani na spletišču spletne

trgovine in ustvarja poročila o uspešnosti spletne trgovine.



Slika 3.13: Nadzorna plošča spletne trgovine.

3.6 Postavitev v produkcijsko okolje

Spletna trgovina je v produkcijskem okolju postavljena na operacijskem sistemu Linux, distribucije in verzije Ubuntu Xenial 16.04 LTS. Pri tem smo uporabili strežnik Nginx in vmesnik uWSGI.

3.6.1 Konfiguracija vmesnika uWSGI

Namesto omrežnih vrat so uporabljene vtičnice Unix, saj so vse komponente, ki se med seboj povezujejo, na istem strežniku. Uporaba vtičnic Unix omogoča varnejšo komunikacijo in boljšo zmogljivost [2]. Podatki med strežnikom in aplikacijo se po vtičnici ne prenašajo preko protokola HTTP, ampak preko protokola uWSGI. Protokol uWSGI ponuja preprost in univerzalen vmesnik med spletnim strežnikom in spletno aplikacijo za ogrodja jezika

Python [16]. Strežnik Nginx privzeto posreduje promet protokola uWSGI, zato je za spletno trgovino najboljša izbira.

Prvi korak konfiguracije vmesnika uWSGI je kreacija datoteke *trgovina.ini* v direktoriju */etc/uwsgi/apps-available*, ki služi kot konfiguracijska datoteka, katere vsebina je prikazana v listingu 3.20. Da protokol uWSGI uporablja spodnjo konfiguracijsko datoteko, je potrebno simbolično povezavo na konfiguracijsko datoteko kopirati v direktorij */etc/uwsgi/apps-enabled*.

Listing 3.20: trgovina.ini

```
[uwsgi]
project = pisarna
base = /var/www/
chdir = %(base)/%(project)
uid = web

home = /home/%(uid)/Env/trgovina
module = wsgi:application

processes = 2
threads = 10

log-x-forwarded-for = true

# Socket configuration
socket = /run/uwsgi/trgovina.sock
chown-socket = %(uid):www-data
chmod-socket = 660
vacuum = true

# PID file
pidfile = /tmp/trgovina.pid

# Kill requests after 30 seconds
harakiri = 30
harakiri-verbose = true

# Custom headers
```

```
add-header = X-Content-Type-Options: nosniff
add-header = X-XSS-Protection: 1; mode=block
add-header = Strict-Transport-Security: max-age=16070400
add-header = Connection: Keep-Alive
add-header = X-Robots-Tag: noindex

; if the client supports gzip encoding goto to the gzipper
route-if = contains:${HTTP_ACCEPT_ENCODING};gzip goto:_gzip
route-run = last:

; pass the response to the gzip transformation
route-label = _gzip
route-run = gzip:
route-run = chunked:
route-run = last:
```

Vmesnik uWSGI preko konfiguracijske datoteke najde datoteko *wsgi.py* v direktoriju aplikacije, katere vsebina je prikazana v listingu 3.32.

Listing 3.21: wsgi.py

```
import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "pisarnaShop.
    settings")

application = get_wsgi_application()
```

Datoteka *wsgi.py* služi za zagon spletne trgovine.

3.6.2 Konfiguracija spletne aplikacije

Pri postavitvi spletne trgovine v produkcijsko okolje je prvi korak pridobitev izvorne kode spletne trgovine, ki se jo naloži z repozitorija GitLab z naslednjim ukazom:

```
git clone git@gitlab.com:matoxxx/pisarna.git
```

Sledi uvoz podatkovne baze *db.json* testnega okolja, ki smo jo pred tem izvozili v datoteko, z ukazom:

```
python manage.py loaddata db.json
```

Za pravilno streženje statičnih datotek je potrebno datoteke pripraviti z ukazom:

```
python manage.py collectstatic
```

V zadnjem koraku konfiguracije je potrebno vnesti pravilne nastavitve v *settings_local.py*, ki se jih ne shranjuje v repozitorij kode in so namenjene delovanju posamične spletne aplikacije. V tej datoteki se nahajajo konfiguracije za dostop do podatkovne baze, poštnega strežnika ter nekatere dodatne nastavitve.

```
DEBUG = False
ALLOWED_HOSTS = ['trgovina.matox.si']
ADMINS = [('Matej␣Hocevar', 'info@matox.si')]
```

V dodatnih nastavitvah, ki so prikazane zgoraj, se tako najdejo ukazi za izklop razhroščevanja spletne trgovine in s tem prikaza napak, ki se zgodijo pri delovanju aplikacije. V primeru, da se zgodi napaka na strani aplikacije, bodo administratorji, ki so navedeni v seznamu *ADMINS*, prejeli elektronsko sporočilo z napako.

3.6.3 Konfiguracija strežnika Nginx

Nginx je spletni strežnik, ki deluje kot obrnjen posrednik za protokola transportne plasti TCP, UDP in aplikacijske plasti HTTP, HTTPS ter druge protokole na aplikacijski plasti [8]. Podobno kot pri konfiguraciji vmesnika uWSGI je potrebna konfiguracija za strežnik Nginx. Datoteka *trgovina*, ki se nahaja v direktoriju */etc/nginx/sites-available*, prikazana v listingu 3.22, opisuje, kako naj se obnaša strežnik, ko dobi zahtevek na določena vrata.

Listing 3.22: Vsebina datoteke */etc/nginx/sites-available/trgovina*.

```
server {
       listen 80;
```

```
        listen [::]:80;
        server_name trgovina.matox.si www.trgovina.matox.si;
        return 301 https://$server_name$request_uri;

        access_log /var/www/pisarna/pisarnaShop/logs/access.log;
        error_log /var/www/pisarna/pisarnaShop/logs/server_error.log;
}

server {
        # SSL configuration

        listen 443 ssl http2;
        listen [::]:443 ssl http2;
        server_name trgovina.matox.si www.trgovina.matox.si;
        include snippets/ssl-trgovina.matox.si.conf;
        include snippets/ssl-params.conf;

        access_log /var/www/pisarna/pisarnaShop/logs/access.log;
        error_log /var/www/pisarna/pisarnaShop/logs/server_error.log;

        location = /favicon.ico { access_log off; log_not_found off; }
        location /static/ {
                root /var/www/pisarna/pisarnaShop/public;
        }

        location / {
                include uwsgi_params;
                uwsgi_pass unix:/run/uwsgi/trgovina.sock;
        }

        location ~ /.well-known {
                allow all;
        }
}
```

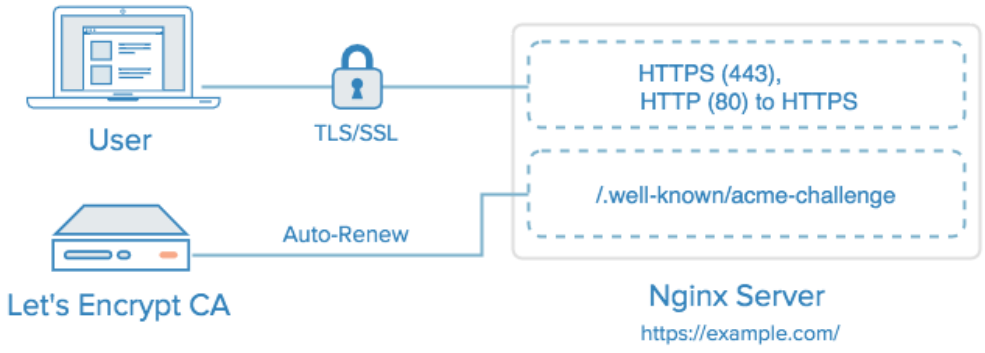
Konfiguracijska datoteka *trgovina* je razdeljena na dva bloka. V prvem bloku *server* so napisana pravila za povezave, ki prihajajo preko vrat 80.

Gre za zahtevke protokola HTTP, ki jih strežnik preusmerja na vrata 433, ki so vrata protokola HTTPS. V drugem bloku so pravila, ki obravnavajo zahtevke preko protokola HTTPS. V tem bloku so opisane lokacije certifikatov SSL, dnevniških datotek in poti do statičnih datotek spletne trgovine. Na koncu datoteke je nastavljena lokacija do vtičnice Unix, ki strežnik Nginx povezuje z vmesnikom uWSGI. Tako kot pri vmesniku uWSGI je potrebno za omogočenje nastavitev strežnika ustvariti simbolično povezavo do datoteke *trgovina* v direktoriju */etc/nginx/sites-enabled*.

3.6.4 Konfiguracija protokola HTTPS

Konfiguracijo protokola HTTPS na strežniku Nginx smo izvedli preko nove certifikatne agencije Let's Encrypt, ki ponuja preprost in brezplačen način za pridobitev in namestitev TLS/SSL certifikatov [2]. Za pridobitev brezplačnega certifikata si je potrebno lastiti domeno, ki preko protokola DNS kaže na naslov IP strežnika spletne trgovine.

Na produkcijskem strežniku preko paketnega upravljalca namestimo paket *letsencrypt*, ki omogoča pridobitev brezplačnega certifikata. Da bo certifikatna agencija uspela prepoznati našo domeno, je potrebno v konfiguraciji strežnika Nginx dovoliti dostop do datoteke */.well-known/acme-challenge*, ki bo odgovorila na izziv programa *letsencrypt* (slika 3.14, spodaj) z naslednjim ukazom:



Slika 3.14: Shema delovanja certifikatne agencije Let's Encrypt s strežnikom Nginx [2].

```
letsencrypt certonly -a webroot --webroot-path=/var/www/html
   -d trgovina.matox.si -d trgovina.matox.si
```

Ob uspešnem preverjanju naše domene bo zgornji ukaz v direktoriju */etc/letsencrypt/archive* ustvaril štiri nove datoteke:

- **cert.pem**: certifikat spletne trgovine,
- **chain.pem**: verižni certifikat agencije Let's Encrypt,
- **fullchain.pem**: združeni datoteki *cert.pem* in *chain.pem*,
- **privkey.pem**: zasebni ključ certifikata spletne trgovine.

Program poleg certifikatov ustvari tudi simbolične povezave do njih v direktoriju */etc/letsencrypt/live/trgovina.matox.si*.

Za povečanje varnosti je potrebno tvoriti močan Diffie-Hellman ključ. S spodnjim ukazom tvorimo 2048 bitni ključ, ki se shrani v direktorij */etc/ssl/certs/dhparam.pem*.

```
openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
```

Sledi kreacija izvlečkov konfiguracijskih datotek, ki bodo kazale na novo ustvarjene certifikate SSL in njihove ključe. V direktoriju */etc/nginx/snippets* ustvarimo datoteko z imenom *ssl-trgovina.matox.si.conf*, katere vsebina je prikazana v listingu 3.23.

Listing 3.23: */etc/nginx/snippets/ssl-trgovina.matox.si.conf*

```
ssl_certificate /etc/letsencrypt/live/trgovina.matox.si/
   fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/trgovina.matox.si/
   privkey.pem;
```

V zadnjem koraku sledi kreacija izvlečka konfiguracije za ključ Diffie-Hellman. V direktoriju */etc/nginx/snippets* ustvarimo datoteko *ssl-params.conf*.

Listing 3.24: */etc/nginx/snippets/ssl-params.conf*

```
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
ssl_prefer_server_ciphers on;
```

```
ssl_ciphers "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH"
    ;
ssl_ecdh_curve secp384r1;
ssl_session_cache shared:SSL:10m;
ssl_session_tickets off;
ssl_stapling on;
ssl_stapling_verify on;
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 5s;
add_header Strict-Transport-Security "max-age=63072000;␣
    includeSubdomains";
add_header X-Frame-Options DENY;
add_header X-Content-Type-Options nosniff;

ssl_dhparam /etc/ssl/certs/dhparam.pem;
```

Konfiguracijski izvleček definira parametre, ki omogočajo visoko varnost na račun slabše združljivosti s starejšimi napravami. Rezultati testa SSL so vidni na sliki v poglavju 3.7.5.

3.6.5 Konfiguracija dinamičnega iskanja

Na začetku je potrebno v konfiguracijski datoteki spletne trgovine *settings.py* konfigurirati nastavitve za iskalni strežnik, kot je prikazano spodaj:

```
HAYSTACK_CONNECTIONS = {
    'default': {
        'ENGINE': 'haystack.backends.solr_backend.SolrEngine'
            ,
        'URL': 'http://127.0.0.1:8983/solr',
        'INCLUDE_SPELLING': True,
    },
}
```

Sledi namestitev paketa Apache Solr različice 4.7.2 preko upravljalca paketov. S spodnjimi ukazi naredimo varnostno kopijo privzete konfiguracije in ustvarimo simbolično povezavo do naše konfiguracije iskalnega strežnika Apache Solr:

```
cd solr-4.7.2/example/solr/collection1
mv conf conf.original
ln -s <webshop_location>/solr conf
```

Ko povežemo konfiguracijsko datoteko z iskalnim strežnikom, se premaknemo po direktorijih nazaj do datoteke *start.jar* in jo poženemo z naslednjim ukazom:

```
cd ../..
java -jar start.jar
```

S tem ukazom smo zagnali iskalni strežnik, ki zna preko konfiguracijske datoteke spletne trgovine komunicirati s trgovino. Zadnji korak je indeksiranje izdelkov v iskalni strežnik. Indeksiranje opravimo s spodnjim ukazom, ki vnese vse naše izdelke in njihove atribute v podatkovno bazo iskalnega strežnika:

```
python manage.py rebuild_index --noinput
>Removing all documents from your index because you said so.
>All documents removed.
>Indexing 201 Products
>Indexing 201 Products
```

3.7 Varnost spletne trgovine

Ko govorimo o varnosti spletne trgovine, se v največji meri zanašamo na varnost ogrodja Django. Ogrodje Django namreč zelo dobro skrbi za varnost in preprečuje večino znanih ranljivosti. V spodnjih razdelkih so predstavljeni najbolj pogosti napadi na spletne aplikacije.

3.7.1 Večdomensko izvajanje kode

Napad večdomenskega izvajanja kode (angl. Cross site scripting, XSS) napadalcu omogoča vstavljanje zlonamerne kode v spletišče. V večini primerov napadalcu uspe kodo shraniti v podatkovno bazo, kjer je vsebina kasneje

prikazana uporabniku. Ob prikazu se zlonamerna koda tudi izvede. Napade večdomenskega izvajanja kode lahko imenujemo tudi napade, ko napadalcu uspe izvesti Javascript kodo v uporabnikovem brskalniku. Napadi večdomenskega izvajanja kode lahko izvirajo iz katerega koli neznanega vira podatkov, ko podatki niso zadostno očiščeni pred uporabo na spletni strani [5]. Uporaba predlog ogrodja Django omogoča preverjanje znakov v spremenljivkah, ki so posebej nevarni pri uporabi na odjemalčevi strani. Ustrezne predloge prevedejo nevarne znake tako, da posebni znaki ne morejo tvoriti zlonamerne programske kode, kljub temu pa poskrbijo za ustrezen prikaz na strani uporabnika.

3.7.2 Vrivanje kode SQL

Vrivanje kode SQL (angl. SQL injection) je tip napada, kjer napadalcu uspe izvesti kodo SQL nad podatkovno bazo. Posledica napada je lahko manipulacija podatkov ali izvoz celotne podatkovne baze [5]. Z uporabo podatkovnih poizvedb ogrodje Django očisti poizvedbo in jo posreduje gonilniku podatkovne baze.

3.7.3 Večdomensko ponarejanje zahtevka

Večdomensko ponarejanje zahtevka (angl. Cross site request forgery, CSRF) omogoča napadalcu izvajanje akcij z uporabo poverilnic drugega uporabnika brez njegove vednosti ali soglasja [5]. Ogrodje Django ima vgrajeno zaščito za večino tipov napadov. Zaščita proti večdomenskemu ponarejanju zahtevkov deluje s preverjanjem žetona, ki se pošilja z vsakim zahtevkom HTTP tipa POST. Zaščita zagotavlja, da napadalec ne more ponoviti zahtevka POST z drugim uporabnikom, saj bi za uspešno izvajanje akcije moral poznati žeton drugega uporabnika.

Zaščita za večdomensko ponarejanje zahtevka je v spletni trgovini omogočena z navedbo *'django.middleware.csrf.CsrfViewMiddleware'* v seznamu vmesne opreme *MIDDLEWARE_CLASSES* v konfiguracijski datoteki *settings.py* (v

listingu 3.25).

Listing 3.25: *settings.py*

```
MIDDLEWARE_CLASSES = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware'
        ,
    'django.contrib.auth.middleware.
        SessionAuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'oscar.apps.basket.middleware.BasketMiddleware',
    'django.contrib.flatpages.middleware.
        FlatpageFallbackMiddleware',
]
```

3.7.4 Clickjacking

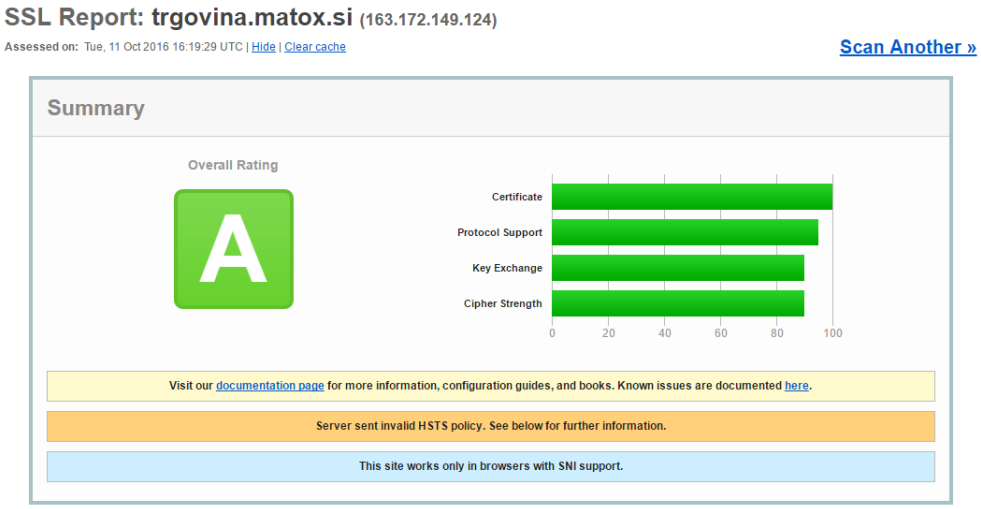
Napad Clickjacking je vrsta napada, kjer zlonamerna spletna stran zavije drugo stran v okvir. Namen napada je prevarati uporabnika v izvedbo neželene akcije na spletni strani [5]. Ogrodje Django vsebuje zaščito pred napadom z uporabo vmesne opreme (angl. middleware) X-Frame-Options, ki je preko opcije *'django.middleware.clickjacking.XFrameOptionsMiddleware'* dodana v konfiguracijsko datoteko *settings.py* v seznam *MIDDLEWARE_CLASSES* (v listingu 3.25). Ogrodje Django s tem prepreči strani, da bi se naložila v okvirju neke druge strani.

V konfiguraciji spletne trgovine je definirano, na katerih domenah je dovoljeno ogrodju Django odgovarjati na zahtevke (v kodi spodaj). Aplikacija ne bo servirala zahtevkov, če prihajajo iz katere koli druge domene, kot je definirana spodaj:

```
ALLOWED_HOSTS = ['trgovina.matox.si']
```

3.7.5 SSL/HTTPS

Komunikacija med odjemalčevo in strežniško stranjo poteka preko tuneliranega protokola SSL/TLS (HTTPS). Protokol SSL/HTTPS zagotavlja, da zlonamerni uporabniki ne morejo prisluškovati prometu, ki poteka med odjemalčevo in strežniško stranjo [5]. Uporabljen je certifikat združenja Let's Encrypt, ki se zavzema, da bi spletne strani uporabljale varne povezave [6]. Konfiguracija strežnika Nginx zagotavlja, da uporabniki spletne trgovine vedno uporabljajo protokol HTTPS. Slika 3.15 prikazuje rezultat testa varnosti razvite spletne strani. S slike lahko razberemo, da je stran ocenjena kot varna, saj ima visoke ocene na vseh testih, ki so bili opravljeni.



Slika 3.15: Preizkus varnosti protokola SSL, ki ga uporablja spletna trgovina preko spletne storitve ssllabs.com.

Poglavje 4

Implementacija priporočilnega sistema

V naslednjih podpoglavjih opisujemo načrt in implementacijo programskega vmesnika, ki strankam spletne trgovine ponuja predloge pri naslednjem nakupu. Izvedba priporočilnega sistema obsega načrt in prototip, implementacijo, opis težav pri implementaciji in analizo priporočilnega sistema.

4.1 Izdelava načrta in prototipa

Želimo izdelati priporočilni sistem, ki bo obstoječim strankam spletne trgovine predlagal izdelke pri naslednjem nakupu glede na njihove prejšnje nakupe ter zraven priporočal dodatne izdelke, ki jih stranka še nikoli ni kupila in bi jo utegnili zanimati.

Prednosti takega priporočilnega sistema so naslednje:

- naročanje je hitro in preprosto,
- stranka ne pozabi naročiti posameznih izdelkov – naroči vse, kar potrebuje,
- stranka se privadi takšnemu načinu nakupovanja in ne išče drugih ponudnikov, ki ne omogočajo podobnega pripomočka,

- če stranka enkrat naroči izdelek, ga bo verjetno naročala tudi v prihodnje,

- s primerno promocijo se lahko poenoti prodajne programe.

Prototip priporočilnega sistema vključuje tabelo z izdelki, za katere je najverjetneje, da jih bo stranka ponovno kupila. Izbirnik (na sliki 4.1 desno zgoraj) omogoča izbiro obdobja, za katerega so izdelki prikazani. Tabela vsebuje sliko, naziv in ceno izdelka, število naročenih izdelkov v izbranem obdobju, podatke o zadnjem naročilu, vnosno polje za novo naročilo in gumb *V košarico*.

Predlagani izdelki

Obdobje: 6 mesecev

Naziv	Cena (z DDV)	V zadnjem obd. naročeno	Zadnje naročilo Datum	Št.		
RIBSON EPSON 7753 Je prozoren in namenjen lepljenju...	7.53€	20	16.6.2016	10	10	
LEPILNI TRAK 15x3 AERO Izdelan iz PVC folije na katero je...	2.61€	50	16.6.2016	50	50	
LEPILNI TRAK 17x03 PRO Osnova je tulec standardne velikosti...	2.85€	108	16.6.2016	55	53	
SPONKE PALIČNE No. 1 Mapa je izdelana iz kvalitetnega PVC...	0.93€	1	23.1.2016	1	1	
SPONKE PALIČNE 24/8 Sprednja platnica je prozorna debeline...	1.05€		Predlagani izdelek		1	
KUVERTA AMERIKANKA Izdelana je iz belega papirja 90gr...	1.00€	1000	3.4.2016	1000	1000	
KUVERTA MODRA Uporabljajo se za pošiljanje dopisov...	0.95€	300	3.4.2016	300	300	
PISALO MARKER Univerzalen model z aluminijastim...	0.97€	12	2.4.2016	10	2	

Slika 4.1: Prototip izgleda priporočilnega sistema v spletni trgovini.

Poleg dosedanjih nakupov bomo stranki v tabeli ponudili tudi nov izdelek, ki ga do sedaj še ni kupila in bi jo utegnil zanimati.

4.2 Pridobivanje in uvoz podatkov o naročilih

Podatke o naročilih strank smo dobili v datoteki s formatom CSV, ki je vsebovala spodnje stolpce:

- STEVILKA – številka naročila,
- DATUM - datum naročila,
- SIFRA_KUPCA – šifra kupca, ki je enaka davčni številki,
- NAZIV_KUPCA - naziv podjetja,
- SIFRA_IZDELKA – kataloška številka izdelka,
- KOLICINA – prodana količina za posamezno naročilo.

Iz dane datoteke smo izluščili podatke o naročnikih in posameznih naročilih ter jih preko implementiranih uvoznikov uvozili v podatkovno bazo spletne trgovine. Na podlagi teh podatkov smo dobili zgodovino naročil pred postavitvijo spletne trgovine ter informacije o naročnikih.

4.2.1 Implementacija uvoznika obstoječih strank

Uvoznika obstoječih strank smo implementirali s pomočjo ukaznega sistema ogrodja Django (angl. Django command management system). Za implementacijo aplikacijskega ukaza je potrebno v direktoriju spletne trgovine *pisarna/management/commands* ustvariti novo datoteko, ki vsebuje razred *Command*. Ogrodje Django ukaz prepozna in ga doda v svoj nabor ukazov. Tako je ukaz mogoče sprožiti preko aplikacijskih ukazov spletne trgovine z ukazom *python manage.py command_name*.

Ustvarili smo novo datoteko *shop_import_customers.py*, v kateri smo definirali ukaz, ki prebere obstoječe stranke iz datoteke v formatu CSV in jih zapiše v podatkovno bazo spletne trgovine. Ukaz za uvoz bomo predstavili v več delih.

V listingu 4.1 spodaj definiramo razred *Command*. V prvi vrstici razreda deklariramo spremenljivko *help*, ki nam prikaže pomoč pri izpisu ukaza z argumentom *–help*. Sledijo deklaracije privzetih vrednosti uvoza, ki se jih prepiše v primeru, da jih podamo preko argumentov.

Listing 4.1: Definicija razreda *Command* v datoteki *shop_import_customers.py*.

```
class Command(BaseCommand):
    help = 'Import customers to the web shop.'
    _prefix = 'imported_'
    _delimiter = ','
    _encoding = 'ISO-8859-2'
```

S kodo v listingu 4.2 spodaj definiramo vse argumente, ki jih sprejme ukaz. Argumenti, ki so definirani za dani ukaz:

- ***file_path***: obvezen argument, v katerem podamo pot od datoteke s podatki,
- ***–delimiter***: znak, ki določa ločilo med posameznimi vrsitcami v datoteki CSV,
- ***–encoding***: nastavitev kodiranja uvozne datoteke,
- ***–flush***: Pred uvozom se izbrišejo vsi obstoječi uporabniki, ki so bili naloženi preko tega uvoznika.

Listing 4.2: Deklaracija argumentov, ki jih sprejme ukaz *shop_import_customers*.

```
    def add_arguments(self, parser):
        parser.add_argument('file_path', type=str)
        parser.add_argument(
            '--delimiter',
            action='store',
            dest='delimiter',
            default=',',
```

```
            help='Set delimiter for an imported file',
        )
        parser.add_argument(
            '--encoding',
            action='store',
            dest='encoding',
            default='ISO-8859-2',
            help='Set encoding for an imported file',
        )
        parser.add_argument(
            '--flush',
            action='store_true',
            dest='flush',
            default=False,
            help='Flush out customer models before the import
                ',
        )
```

Metoda *handle* je podedovana metoda razreda *BaseCommand* in mora biti implementirana za uspešno delovanje ukaza. V tej metodi se nastavijo morebitni argumenti ukaza in klic logike, ki skrbi za izvedbo želenega ukaza.

V listingu 4.3 je definicija metode *handle*, v kateri se poleg nastavitve argumentov kliče tudi metoda *_import*.

Listing 4.3: Definicija metode *handle*, ki narekuje izvedbo ukaza.

```
    def handle(self, *args, **options):
        try:
            file_path = options['file_path']
            Validator().validate(file_path)

            if options['flush']:
                self.stdout.write(' - Flushing customers data
                    before the import')
                self._flush_customers_data()

            if options['delimiter']:
                self._delimiter = options['delimiter']
```

```
            if options['encoding']:
                self._encoding = options['encoding']

            self.stdout.write(' - Importing new data')
            self._import(file_path)
        except ImportingError:
            raise ImportingError(_('No file path supplied'))
        except:
            raise ImportingError(_('Error while importing
                data'))
```

Metoda *handle* metodi *_import* posreduje pot do datoteke, ki je bila podana kot argument ukaza. Kot je prikazano v listingu 4.4 spodaj, metoda prebere podano datoteko in se sprehodi čez vse vrstice te datoteke. Vsako vrstico pošlje metodi *_create_customer*. Po koncu branja datoteke nam ukaz izpiše statistiko vnesenih strank.

Listing 4.4: Metoda *_import* se sprehodi preko prebranih vrstic podane datoteke in poskrbi za uvoz strank.

```
    def _import(self, file_path=None):
        stats = {'new_customers': 0, 'updated_customers': 0}

        with open(file_path, 'r') as csvfile:
            reader = self.unicode_dict_reader(csvfile,
                delimiter=self._delimiter, lineterminator='\n'
                )
            for row in reader:
                self._create_customer(row, stats)

        msg = "New customers: %d, updated customers: %d" % (
            stats['new_customers'], stats['updated_customers'
            ])
        self.stdout.write(msg)
```

Zadnja metoda, ki smo jo pri uvozniku obstoječih strank implementirali, je metoda *_create_customer*, ki je prikazana v listingu 4.5. Metoda na začetku

iz uvožene vrstice datoteke prebere davčno številko in naziv stranke. Uvoz se nadaljuje s preverjanjem, če stranka že obstaja. V *try-except* bloku se naredi poizvedba v podatkovno bazo, ki poskuša najti uporabnika z danim uporabniškim imenom. V primeru, da uporabnik z danim uporabniškim imenom ne obstaja, se proži izjema *User.DoesNotExist*, ki jo ujamemo v *except* bloku. Ob sprožitvi izjeme vemo, da uporabnik z danim uporabniškim imenom še ne obstaja, zato se lahko v podatkovno bazo shrani nova stranka.

Novo uvoženi stranki se dodeli uporabniško ime, ki je sestavljeno iz predpone *"imported_"* in davčne številke podjetja. Stranki se dodeli naključno geslo in odvzame pravice za direktno uporabo računa. S tem se zagotovi, da se v račun ni mogoče prijaviti ali ga kako drugače uporabljati.

Listing 4.5: Definicija metode *_create_customer*, ki v podatkovno bazo shrani uporabnika.

```
    def _create_customer(self, row, stats):
        vac_number = row['vac_number']
        company_name = row['company_name']
        username = self._prefix + vac_number
        try:
            user = User.objects.get(username=username)
            self._create_address(user, vac_number,
                company_name)
            stats['updated_customers'] += 1
        except User.DoesNotExist:
            user = User(
                username=username,
                password=User.objects.make_random_password(),
                email='%s@temp.si' % username,
                is_active=False,
                is_staff=False,
                is_superuser=False
            )
            user.save()
            self._create_address(user, vac_number,
                company_name)
            stats['new_customers'] += 1
```

4.2.2 Implementacija uvoznika naročil

Podobno kot pri uvozniku obstoječih strank smo nov ukaz implementirali za uvoz vseh naročil. Uvoznik tudi v tem primeru preko datoteke v formatu CSV uvozi podatke v podatkovno bazo.

Ker je struktura ukaza podobna kot pri uvozu obstoječih strank, bomo predstavili samo najzanimivejšo metodo v ukazu: metodo, ki uvozi naročila. Ukaz sprejme naslednje argumente:

- ***file_path***: obvezen argument, v katerem podamo pot od datoteke s podatki,
- ***–delimiter***: znak, ki določa ločilo med posameznimi vrsticami v datoteki CSV,
- ***–encoding***: nastavitev kodiranja uvozne datoteke,
- ***–flush***: pred uvozom se izbrišejo vsi obstoječi uporabniki, ki so bili naloženi preko tega uvoznika.

Metoda *_create_order* preko argumentov prejme vrstico uvožene datoteke z informacijami o naročilu. Kot je prikazano v listingu 4.6, se na začetku metode iz vrstice izlušči podatke o naročilu.

Listing 4.6: Metoda *_create_order*, ki v podatkovno bazo shrani naročilo.

```
    def _create_order(self, row, stats):
        self.basket = factories.create_basket(empty=True)
        order_number = row['order_number']
        raw_quantity = row['quantity']
        vac_number = row['vac_number']
        raw_date = row['date']
        product_id = row['product_id']
```

Metoda za uvoz naročil se nadaljuje z blokom *try-except*, prikazanem spodaj, ki lovi izjeme, ki se pri uvozu lahko zgodijo. Sledi pridobivanje objekta *UserAddres* in *Product*, v katerem sta shranjena naslov stranke in izdelek, ki je bil naročen.

```
        try:
            # Getting UserAddress object
            userAddress = UserAddress.objects.get(vac_number=
                vac_number)
            if userAddress is None:
                raise UserAddress.DoesNotExist()
            if userAddress.user is None:
                raise Product.DoesNotExist()
            # Getting Product object
            try:
                product = Product.objects.get(upc=product_id)
                if product is None:
                    raise Product.DoesNotExist()
            except:
                raise Product.DoesNotExist()
```

Ob pridobitvi objektov iz podatkovnega modela spremenljivko *quantity* nastavimo na številsko vrednost in poskrbimo, da je količina naročenih izdelkov vedno večja od nič. Poskrbimo tudi, da je datum naročila v spremenljivki *date* v želenem formatu.

```
            # Parsing quantity
            quantity = int(raw_quantity)
            if quantity < 1:
                quantity = 1
            # Parsing DateTime object
            date = dateParser.parse(raw_date)
```

V bloku *try-except*, prikazanem spodaj, preverimo, ali gre za obstoječe naročilo. Izdelek dodamo v košarico in oddamo naročilo v podatkovno bazo spletne trgovine.

```
            try:
```

```
            temp_order = Order.objects.get(number=
                order_number)
            stats['updated_orders'] += 1
        except:
            stats['new_orders'] += 1
        add_product(self.basket, product.
            min_child_price_excl_tax, quantity=quantity,
            product=product)
        order = place_order(
            self.creator,
            user=userAddress.user,
            basket=self.basket,
            order_number=order_number,
            shipping_method=PersonalDelivery(),
            status='Complete',
            date_placed=date,
            {'bulk_import': True}
        )
```

Na repu metode (koda spodaj) skrbimo za lovljenje izjem in pravilno posodabljanje statistike uvoza.

```
    except ValueError:
        stats['skipped_orders'] += 1
    except UserAddress.DoesNotExist:
        stats['skipped_customers'] += 1
    except Product.DoesNotExist:
        stats['skipped_orders'] += 1
    except:
        stats['skipped_customers'] += 1
```

4.3 Implementacija priporočilnega sistema

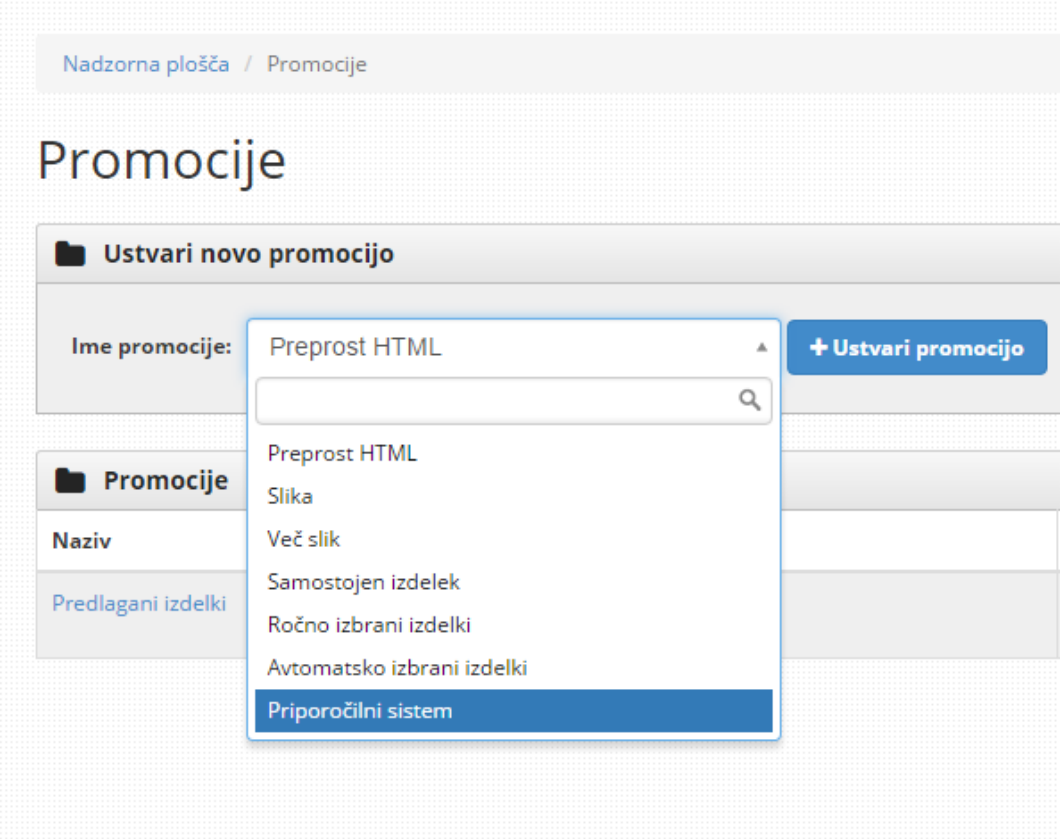
Implementacijo priporočilnega sistema smo ločili na dva dela: na implementacijo seznama predlaganih izdelkov na podlagi prejšnjih nakupov, ki je prikazan kot seznam *Predlogi nakupov*, ter na implementacijo priporočilnega sistema, ki strankam na podlagi prejšnjih nakupov priporoči izdelek, ki ga še

nikoli niso kupile. Priporočeni izdelki so vrinjeni v seznam *Predlogi nakupov* in se od predlaganih izdelkov ločijo po stolpcu *Zadnje naročilo*, v katerem je namesto informacij o zadnjem naročilu besedilo *Priporočen izdelek* (slika 4.1).

4.3.1 Implementacija nastavitev prikaza priporočilnega sistema

Za prikaz predlaganih izdelkov smo uporabili funkcionalnost promocij, ki jih ogrodje Oscar ponuja za prikaz različnih vsebin v spletni trgovini. Ogrodje Oscar ponuja nekaj osnovnih tipov promocij kot so *Preprost HTML*, *Slika*, *Več slik*, *Samostojen izdelek*, *Ročno izbrani izdelki* in *Avtomatsko izbrani izdelki*. Promocije so primerne za prikaz predlaganih izdelkov, saj se lahko eno ali več uporabi na eni ali več straneh hkrati.

Ker nobena od navedenih tipov promocij ni bila ustrezna za prikaz predlaganih izdelkov, smo se odločili, da razširimo nabor promocij s tipom *Priporočilni sistem*. Na sliki 4.2 spodaj je prikazan spustni seznam za kreacijo nove promocije.



Slika 4.2: Kreiranje nove promocije v nadzorni plošči spletne trgovine.

Na sliki 4.3 spodaj je prikazan obrazec, ki se pojavi ob kreiranju nove promocije tipa *Priporočilni sistem*.

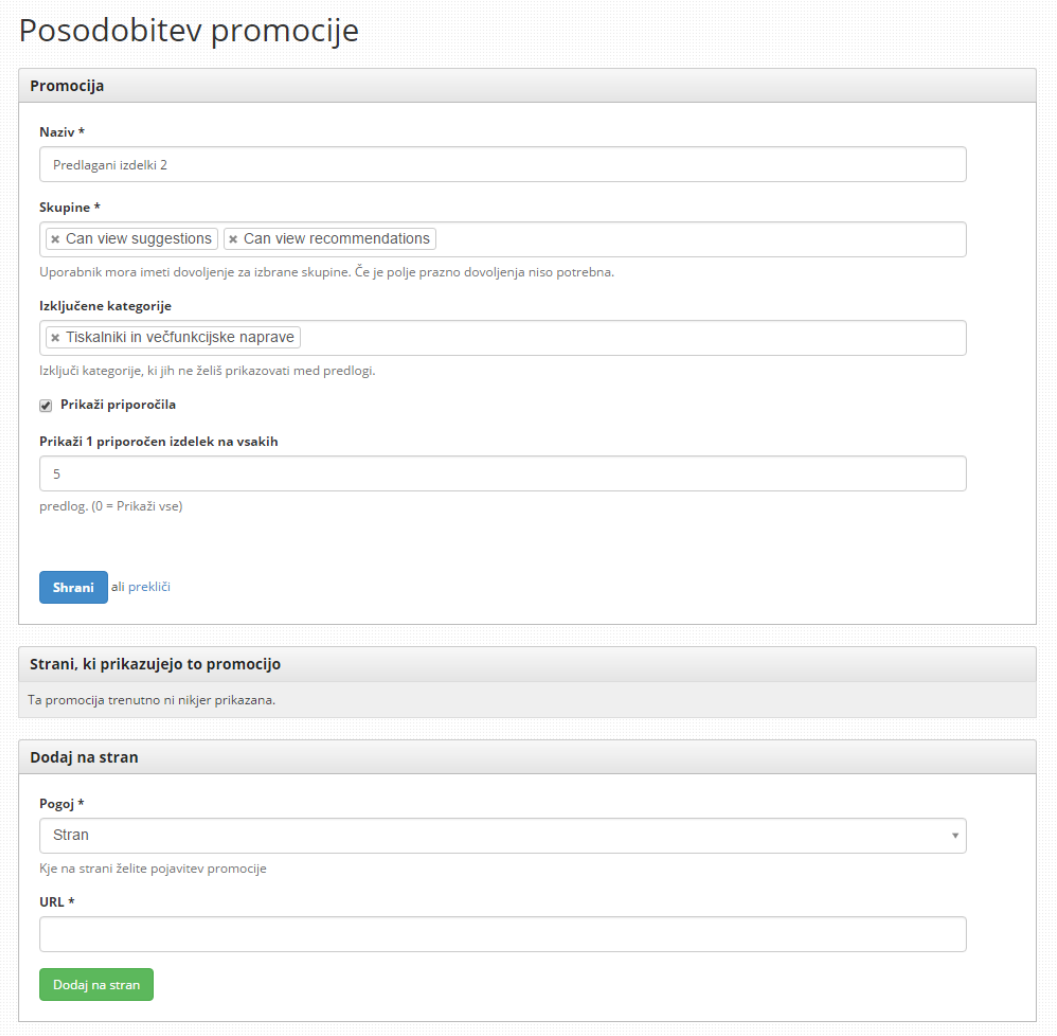
Na začetku mora upravitelj vpisati naziv promocije in skupine pravic, ki jih mora uporabnik imeti za prikaz promocije. Skupine pravic mora upravitelj strani definirati pred kreacijo promocije. Na ta način lahko naredimo podmnožico uporabnikov, ki imajo pravice do prikaza priporočilnega sistema. V primeru promocije priporočilnega sistema je smiselno prikazovati promocijo vsem uporabnikom, ki imajo pravice za ogled predlogov (*Can view suggestions*) in pravice za ogled priporočil (*Can view recommendations*).

Sledi izbirno polje kategorij, ki jih upravitelj izključi s seznama predlaganih in priporočenih izdelkov. Opisano polje je uporabno takrat, ko upravitelj

ne želi prikazovati izdelkov iz določenih kategorij.

Na koncu obrazca sta potrditveno polje *Prikaži priporočila* in vnosno polje, s katerim se lahko nastavi, kolikšen delež predlaganih izdelkov so priporočeni izdelki. S potrditvenim poljem ima upravitelj možnost, da izklopi prikaz priporočenih izdelkov v seznamu predlaganih izdelkov. Tako se uporabnikom v seznamu predlaganih izdelkov ne bodo prikazovali izdelki, ki jih še nikoli niso naročili.

S pritiskom na gumb *Shrani* se obrazec za urejanje promocije posodobi, spodaj pa se upravitelju prikaže razdelek *Dodaj na stran*, v katerega lahko vpišemo, na katerih naslovih URL želimo prikazovati na novo ustvarjeno promocijo. S shranitvijo nastavitev se na nastavljeni strani prikaže željena promocija.



Slika 4.3: Obrazec za kreiranje nove promocije tipa *Priporočilni sistem* v nadzorni plošči spletne trgovine.

4.3.2 Implementacija seznama predlaganih izdelkov

V spodnjih razdelkih je predstavljana implementacija seznama predlaganih izdelkov, ki smo jo razdelili na podatkovni model in sistem za določitev predlaganih izdelkov.

Podatkovni model predlaganih izdelkov

V Pythonovskem razredu, ki je prikazan v listingu 4.7, je definiran model predlaganih izdelkov. Vsak predlagani izdelek vključuje polja izdelek, uporabnik in število nakupov v določenem obdobju. S pomočjo teh polj se lahko uporabniku prikaže tabela predlaganih izdelkov v spletni trgovini.

Listing 4.7: Definicija podatkovnega modela predlaganih izdelkov.

```
class Suggestion(models.Model):
    product = models.ForeignKey(Product, verbose_name=_('
        Product'))
    user = models.ForeignKey(AUTH_USER_MODEL, verbose_name=_(
        'User'), related_name='suggestions')
    num_purchases = models.PositiveIntegerField(verbose_name=
        _('Number of purchased'), default=1)
```

Sistem predlaganih izdelkov

Sistem predlaganih izdelkov, prikazan v listingu 4.8, je definiran kot razred. Ob inicializaciji metode se nastavi začetek obdobja, v kateri želimo določati predlagane izdelke.

Listing 4.8: Implementacija sistema predlaganih izdelkov.

```
class SuggestionSystem:
    period = date(2016, 6, 1)
    def __init__(self, period=None):
        if period:
            self.period = period
        else:
            self.period = getattr(settings, '
                SHOP_RECOMMENDATIONS_FROM_DATE', self.period)
```

Spodaj prikazana metoda *get_suggestions* razreda *SuggestionSystem* vrača vsa priporočila uporabnika. Uporabnik je podan kot argument metode.

```
    @staticmethod
    def get_suggestions(user):
        return Suggestion.objects.filter(user=user)
```

Metoda *calculate_suggestions*, ki je prikazana spodaj, predstavlja jedro sistema predlaganih izdelkov. V tej metodi se določijo izdelki, ki so primerni za predlaganje in se preko objekta *Suggestion* shranijo v podatkovno bazo. Metoda se v prvi zanki sprehodi po vseh naročilih, ki so bila novejša od nastavljenega začetka obdobja. V notranji zanki se sprehodi čez vse postavke naročila in jih doda v objekt *Suggestion*, ki se na koncu zanke shrani. Predlogu se doda uporabnik, izdelek in količina naročil izdelka. Blok *try-except* skrbi, da se primerno obravnava izjemo, če ta predlog še ne obstaja. V primeru izjeme se doda nov predlog v podatkovno bazo. Podrobnejša analiza opisane metode se nahaja v naslednjem, 5. poglavju.

```
    def calculate_suggestions(self):
        stats = {'new_suggestions': 0, '
            incremented_suggestions': 0, 'total_suggestions':
            0}
        for order in Order.objects.filter(date_placed__gte=
            self.period):
             for line in order.lines.all():
                try:
                    suggestion = Suggestion.objects.get(user=
                        order.user, product=line.product)
                    suggestion.num_purchases += line.quantity
                    stats['incremented_suggestions'] += 1
                except Suggestion.DoesNotExist:
                    suggestion = Suggestion(user=order.user,
                                            product=line.
                                                product,
                                            num_purchases=
                                                line.quantity)
                    stats['new_suggestions'] += 1
                stats['total_suggestions'] += 1
                suggestion.save()
        return stats
```

Določitev predlaganih izdelkov

Za določitev predlaganih izdelkov smo, podobno kot pri uvozu strank in obstoječih naročil, uporabili ukazni sistem ogrodja Django. V listingu 4.9 spodaj je definirana metoda *_calculate_suggestions*, ki predstavlja jedro ukaza *shop_suggestions_calculate*. Ukaz, ki je prikazan v listingu 4.9, kliče metodo sistema predlog (angl. template) *calculate_suggestions*, ki določa predloge za dano periodo. Sistem predlog vrača statistiko, ki jo ukaz na koncu izpiše na sistemski izhod. Statistika vsebuje podatke, koliko predlog je bilo ustvarjenih na novo, koliko je bilo posodobljenih in koliko predlog je novih.

Listing 4.9: Ukaz *shop_suggestions_calculate* za izračun predlaganih izdelkov.

```
    def _calculate_suggestions(self):
        stats = self.suggestion_system.calculate_suggestions
            ()
        msg = "New suggestions: %d, incremented suggestions:
            %d, total_suggestions: %d" % (
            stats['new_suggestions'],
            stats['incremented_suggestions'],
            stats['total_suggestions']
        )
        self.stdout.write(msg)
```

4.3.3 Implementacija seznam priporočenih izdelkov

Podobno kot pri implementaciji predlaganih izdelkov smo tudi predstavitev implementacije priporočenih izdelkov razdelili na podatkovni model in sistem za določitev priporočenih izdelkov. K podatkovnemu modelu je dodan še model, ki omogoča določitev seznama priporočenih izdelkov, ki jih želi priporočiti strankam v priporočilnem sistemu.

Podatkovni model priporočenih izdelkov

Definicija podatkovnega modela *RecommendationList*, ki je prikazana v listingu 4.10, predstavlja definicijo priporočilnih izdelkov, ki jih upravitelj iz-

bere za prikaz v seznamu predlaganih izdelkov. Model *RecommendationList* povezuje polja kategorija, izdelek, prioriteta, ki predstavljajo pomembnost priporočila, oznako ali je priporočilo javno ter čas kreiranja.

Listing 4.10: Definicija podatkovnega modela seznama priporočenih izdelkov.

```
class RecommendationList(models.Model):
    category = models.ForeignKey(Category, verbose_name=_('
        Category'), help_text=_('Select category for
        recommendation calculation.'))
    product = models.ForeignKey(Product, verbose_name=_('
        Product'), help_text=_('Select recommended product
        from selected category.'))
    priority = models.PositiveSmallIntegerField(verbose_name=
        _('Priority'), blank=True, null=True, unique=True,
        default=0, help_text=_('Select priority of
        recommendation, higher priority means higher
        importance.'))
    is_public = models.BooleanField(_('Is public?'), default=
        False, help_text=_("Public recommendations have a
        customer-facing page"))
    date_created = models.DateTimeField(_("Date Created"),
        auto_now_add=True)
```

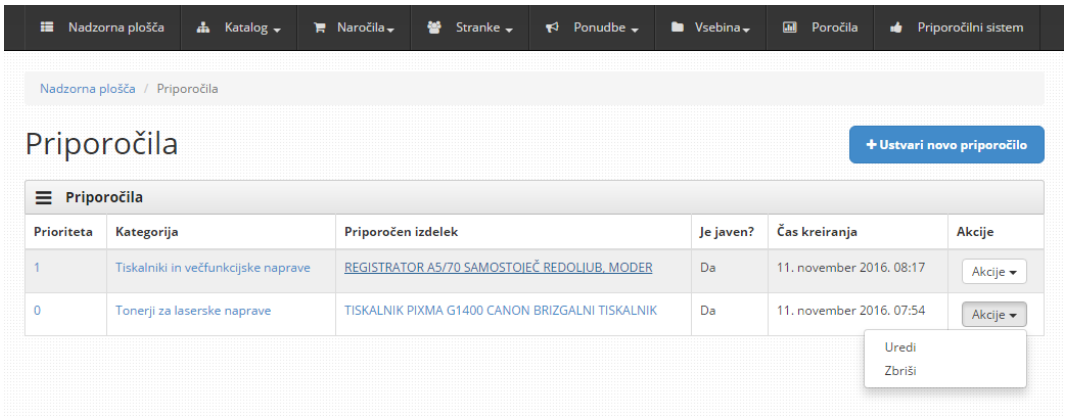
Zadnji podatkovni model, ki smo ga definirali, je model *Recommendation*, prikazan v listingu 4.11. Model je namenjen povezovanju modelov uporabnika in podatkovnega modela *RecommendationList*, opisanega v listingu 4.10.

Listing 4.11: Podatkovni model priporočenih izdelkov.

```
class Recommendation(models.Model):
    user = models.ForeignKey(AUTH_USER_MODEL, verbose_name=_(
        'User'), related_name='recommendations')
    recommendation = models.ForeignKey(RecommendationList,
        verbose_name=_('Recommendation'))
```

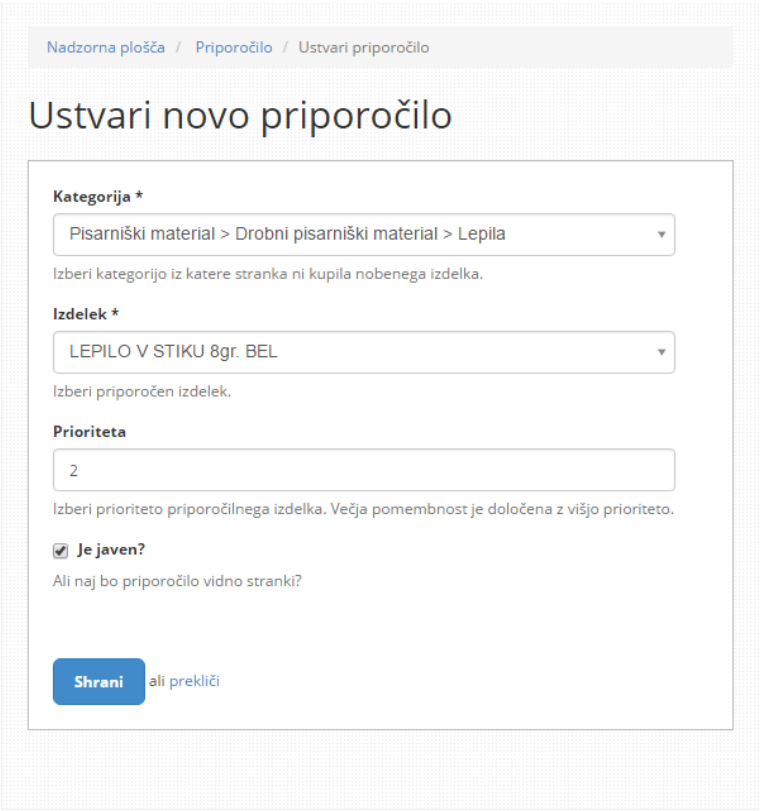
Definicija priporočenih izdelkov

Upravitelju smo v meni nadzorne plošče dodali nov zavihek *Priporočilni sistem*, na sliki 4.4 zgoraj desno, ki mu ponudi možnost dodajanja priporočil. Tu lahko upravitelj definira, katere izdelke priporoča strankam, če med svojimi naročili še niso kupile izdelka iz določene kategorije. Na zavihku *Priporočilni sistem* se upravitelju prikaže tabela vseh priporočil, ki jih je ustvaril. Upravitelju so omogočene akcije pregleda, dodajanja, urejanja ali brisanja priporočil.



Slika 4.4: Seznam priporočil, ki jih nastavi upravitelj preko nadzorne plošče.

S pritiskom na gumb *Ustvari novo priporočilo* se upravitelju odpre nov obrazec, kjer lahko ustvari novo priporočilo. Kot je prikazano na sliki 4.5, mora upravitelj v začetku vpisati kategorijo, iz katere stranka ni kupila nobenega izdelka. Nato sledi izbirnik za priporočen izdelek. V vnosnem polju pod izbirnikom priporočenega izdelka upravitelj določi prioriteto upoštevanja novega priporočila. Višja kot je nastavljena številka, višja je prioriteta. Prioriteta je pomembna pri prikazovanju priporočenih izdelkov v seznamu predlaganih izdelkov. Zadnji element obrazca je potrditveno polje, ki nastavi vidnost priporočila v spletni trgovini.



Slika 4.5: Obrazec za kreacijo novega priporočila v spletni trgovini.

Sistem priporočenih izdelkov

Podobno kot sistem predlaganih izdelkov je tudi sistem priporočenih izdelkov definiran kot razred. V inicializaciji razreda se nastavi časovno obdobje, za katerega želimo določiti priporočene izdelke. V listingu 4.12 je prikazana koda razreda *RecommendationSystem*.

Listing 4.12: Implementacija sistema priporočenih izdelkov.

```
class RecommendationSystem:

    period = date(2016, 6, 1)

    def __init__(self, period=None):
        if period:
            self.period = period
```

```
        else:
            self.period = getattr(settings, '
                SHOP_RECOMMENDATIONS_FROM_DATE', self.period)
```

V spodnji kodi je metoda *get_recommendations*, ki sprejme kot argument uporabnika in število priporočil, ki jih želimo kot rezultat metode. Metoda poišče vsa priporočila za danega uporabnika in jih razvrsti po prioriteti. Rezultat metode je seznam priporočil z najvišjo prioriteto.

```
    @staticmethod
    def get_recommendations(user, num=1):
        recommendations = Recommendation.objects.filter(user=
            user).order_by('recommendation__priority')
        if num > 0:
            return recommendations[:num]
        else:
            return recommendations
```

Glavno jedro razreda *RecommendationSystem* je v metodi *calculate_recommendations*, prikazani spodaj, ki na podlagi naročil in seznama priporočil, ki ga je ustvaril upravitelj, določi priporočila za vsakega uporabnika posebej.

```
    def calculate_recommendations(self):
        stats = {'new_recommendations': 0, '
            skipped_recommendations': 0, '
            total_recommendations': 0}
        for order in Order.objects.filter(date_placed__gte=
            self.period):
            for line in order.lines.all():
                for r in RecommendationList.objects.all():
                    recommendation_path = r.product.
                        get_categories().first().path
                    product_path = line.product.
                        get_categories().first().path
                    if product_path not in
                        recommendation_path:
                        try:
                            recommendation = Recommendation.
                                objects.get(user=order.user,
```

```
                                recommendation=r)
                            stats['skipped_recommendations']
                                += 1
                        except Recommendation.DoesNotExist:
                            recommendation = Recommendation(
                                user=order.user,
                                recommendation=r)
                            stats['new_recommendations'] += 1
                        recommendation.save()
                        stats['total_recommendations'] += 1
        return stats
```

Določitev priporočenih izdelkov

Tako kot pri določitvi predlaganih izdelkov je izračun priporočenih izdelkov ustvarjen preko ukaznega sistema ogrodja Django. Definirali smo ukaz *shop_recommendations_calculate*, prikazanega v listingu 4.13, ki na podlagi podanih argumentov sproži določanje priporočenih izdelkov. Ukaz kliče metodo *calculate_suggestions* razreda *SuggestionSystem*. Zgoraj opisana metoda *calculate_suggestions* vrača statistiko določanja, ki jo ukaz prejme in izpiše na sistemski izhod.

Listing 4.13: Jedro metode ukaza *shop_recommendations_calculate*, ki določa priporočila za spletno trgovino.

```
    def _calculate_recommendations(self):
        stats = self.recommendation_system.
            calculate_recommendations()
        msg = "New recommendations: %d, skipped
            recommendations: %d, total recommendations: %d" %
            (
             stats['new_recommendations'],
             stats['skipped_recommendations'],
             stats['total_recommendations']
        )
        self.stdout.write(msg)
```

4.3.4 Implementacija prikaza predlaganih in priporočenih izdelkov

Naslednja faza implementacije je prikaz določenih predlaganih in priporočenih izdelkov strankam kot seznam v spletni trgovini. V ta namen smo se poslužili funkcionalnosti oznak predlog (angl. template tags), ki jih omogoča ogrodje Django, in so del predlog ogrodja Django. Namen oznak predlog je povezovanje podatkov v predloge, ki servirajo kodo HTML. Spodaj so opisane oznake (angl. template tags), ki smo jih implementirali za namen pridobivanja informacij o predlaganih in priporočenih izdelkih iz podatkovne baze spletne trgovine.

Implementacija oznake za seznam predlaganih izdelkov

Prva oznaka (angl. template tag) se imenuje *suggested_products*. Namen oznake je pridobivanje seznama predlaganih izdelkov. Metoda, ki definira oznako *suggested_products*, je anotirana z anotacijo *@register.assignment_tag* in je prikazana v listingu 4.14. Z anotacijo ogrodje Django prepozna novo oznako, ki se lahko uporablja v predlogah. Oznaka prejme dva argumenta: trenutni zahtevek in promocijo. Oznaka *suggested_products* na začetku preveri, če je stranka iz zahtevka avtenticirana v spletno trgovino.

Listing 4.14: Definicija oznake *suggested_products*, katere namen je pridobivanje seznama predlaganih izdelkov.

```
@register.assignment_tag
def suggested_products(request, promotion):
    suggestions = []
    if request.user.is_authenticated():
```

V naslednjem koraku, prikazanem spodaj, preveri, če ima stranka potrebne pravice za pridobitev predlaganih izdelkov.

```
        # Check if user has all group permissions
        for group in promotion.groups.all():
            if group not in request.user.groups.all():
                return None
```

Sledi pridobitev izključenih kategorij izdelkov in seznama vseh predlogov za stranko v tem zahtevku. Če ni izključenih kategorij, se na tem mestu vrne seznam z vsemi predlaganimi izdelki.

```
        excluded_categories = promotion.excluded_categories.
            all()
        all_suggestions = SuggestionSystem().get_suggestions(
            user=request.user)
        if excluded_categories is None:
            return all_suggestions
```

V zadnjem koraku sledi iteracija čez vse predlagane izdelke, kjer zanka preveri, da v končni nabor predlaganih izdelkov doda samo tiste izdelke, ki niso v izključenih kategorijah, ki jih je nastavil upravitelj.

```
        for suggestion in all_suggestions:
            is_included = False
            for category in suggestion.product.get_categories
                ().all():
                if category in excluded_categories:
                    is_included = True
                    break
            if not is_included:
                suggestions.append(suggestion)
    return suggestions
```

Implementacija oznake za seznam priporočenih izdelkov

Podobno kot je definirana oznaka *suggested_products*, smo definirali tudi oznako *recommended_products*. Oznaka, ki je prikazana v listingu 4.15, poleg argumentov trenutni zahtevek in promocija sprejeme tudi število, ki določa koliko priporočenih izdelkov želimo v rezultatu oznake. Oznaka v svoji kodi preveri, če je stranka v zahtevku avtenticirana ter če promocija omogoča prikaz priporočil v svojem seznamu. S prikazom priporočil upravlja upravitelj pri kreaciji promocije. Sledi klic metode *get_recommendations* razreda *RecommendationSystem* s podanim uporabnikom in želenim številom vrnjenih priporočil.

Listing 4.15: Definicija oznake *recommended_products*, katere namen je pridobivanje seznama priporočenih izdelkov.

```
@register.assignment_tag
def recommended_products(request, promotion, num=1):
    recommendations = None
    if request.user.is_authenticated() and promotion.
        display_recommendations:
         recommendations = RecommendationSystem().
             get_recommendations(user=request.user, num=num)
    return recommendations
```

Implementacija oznake za določitev števila prikazanih priporočil

Zadnja oznaka, ki se uporablja pri prikazu seznama predlaganih in priporočenih izdelkov, je *calc_num_of_recommendations*. Predstavljena je v listingu 4.16. Oznaka je namenjana določitvi želenega števila priporočenih izdelkov, ki ga je upravitelj podal pri kreaciji promocije tipa *Priporočilni sistem*. Število je izračunano na podlagi vrednosti, ki jo je upravitelj vnesel pri promociji. Oznaka kot argument prejme promocijo in seznam predlaganih izdelkov ter vrne izračunano število priporočil.

Listing 4.16: Oznaka *calc_num_of_recommendations* vrača število priporočenih izdelkov, ki jih upravitelj želi prikazati.

```
@register.assignment_tag
def calc_num_of_recommendations(promotion, suggestions):
    if promotion.recommendations_per_suggestions > 0:
        return len(suggestions) / promotion.
            recommendations_per_suggestions
    else:
        return 0
```

Implementacija predloge za prikaz seznama

Predloga, ki je prikazana v listingu 4.17, služi prikazu seznama strankam spletne trgovine. Na začetku se naložijo oznake, ki so uporabljene pri pri-

kazu. Poleg že opisanih oznak se uporabljata tudi oznaki za prikaz valute in internacionalizacije besedil. Sledi preverjanje, če je uporabniški zahtevek avtenticiran, ter nalaganje seznama predlaganih in priporočenih izdelkov. V nadaljevanju je definirana tabela, ki skrbi za pravilen prikaz seznama.

Listing 4.17: Predloga, ki stranki prikazuje seznam predlaganih in priporočenih izdelkov .

```
{% load currency_filters %}
{% load product_tags %}
{% load recommendation_tags %}
{% load i18n %}

{% if user.is_authenticated %}
  {% suggested_products request promotion as suggestions %}
  {% calc_num_of_recommendations promotion suggestions as
     recommendation_num %}
  {% recommended_products request promotion
     num=recommendation_num as recommendations %}

  <div class="container-fluid page">
    <div class="page_inner">
      <article class="well promotion promotion_recommended">
        <h1>{{ promotion.name }}</h1>
        <div class="header-divider"></div>

        <div class="suggestion-box">
          <div class="suggestion-table">
            <div class="table-upper-header row">
              <div class="col-md-2 col-md-offset-9
                 text-center">
                <p>{% trans "Last order" %}</p>
              </div>
            </div>
            <div class="table-header row">
              <div class="col-md-1"></div>
              <div class="col-md-5"><p>{% trans "Title" %}</p
                 ></div>
```

```
            <div class="col-md-1"><p>{% trans "Price incl.
                Tax" %}</p></div>
            <div class="col-md-2"><p class="text-center">{%
                trans "In last period ordered" %}</p></div>
            <div class="col-md-2 row text-center">
              <div class="col-md-6 highlighted"><p>{% trans
                  "date" %}</p></div>
              <div class="col-md-6 highlighted"><p>{% trans
                  "quantity" %}</p></div>
            </div>
            <div class="col-md-1"></div>
          </div>
          <div class="table-body">
            {% for suggestion in suggestions %}
              {% include 'promotions/partials/
                 suggestion.html' with
                 product=suggestion.product
                 recommendation=False %}
            {% endfor %}
            {% for recommendation in recommendations %}
              {% include 'promotions/partials/
                 suggestion.html' with
                 product=recommendation.recommendation.product
                  recommendation=True %}
            {% endfor %}
          </div>
          <div class="table-footer">
            <button type="submit" class="btn btn-lg
               btn-primary btn-add-to-basket" value="{%
               trans "Add to basket" %}" data-loading-text=
               "{% trans 'Adding...' %}">
              <i class="icon-shopping-cart"></i>
              {% trans "Add to basket" %}
            </button>
          </div>
        </div>
      </div>
    </article>
```

```
    </div>
  </div>
{% endif %}
```

Končni izgled seznama predlaganih in priporočenih izdelkov je prikazan na sliki 4.6. Stranki se prikaže seznam rednih naročil, ki jih opravlja v spletni trgovini. Na voljo ima informacije o zadnjem naročilu in količini nekega izdelka, ki ga je naročila v zadnjem obdobju. Na skrajni desni strani se nahaja vnosno polje, kjer lahko stranka izpolni svoje naročilo in doda dodatne izdelke v svojo košarico. Na koncu seznama se pripnejo še priporočeni izdelki, ki bi stranko utegnili zanimati. Pod seznamom se na desni strani nahaja gumb za oddajo skupnega naročila.

Slika 4.6: Končni izgled promocije tipa *Priporočilni sistem* kot seznam predlaganih izdelkov v spletni trgovini.

Poglavje 5

Analiza

V tem poglavju analiziramo novo ustvarjeni priporočilni sistem. Predstavljamo tudi težave, ki so se ob implementaciji pojavile, ter analiziramo časovno zahtevnost določitve predlaganih in priporočenih izdelkov. Za konec predstavljamo še možne spremembe, ki bi spletno trgovino lahko izboljšale.

5.1 Predstavitev težav

Prva težava, ki se pojavlja vseskozi ob uporabi sistema, je določitev predlaganih in priporočenih izdelkov. Trenutna rešitev sproži izračunavanje vsak dan, neodvisno od števila nakupov in strank. Priporočilni sistem lahko tako prikaže tudi do 24 ur stare predloge.

Druga težava je v tem, da se algoritem vedno sprehodi čez vsa naročila, neodvisno od tega, če so priporočeni izdelki že določeni v podatkovni bazi. Algoritem tako zapravlja sredstva za izračunavanje, ki jih je že opravil. Izvajanje priporočilnega sistema bi bilo mogoče prilagoditi tako, da določa predloge za točno določenega uporabnika.

5.2 Analiza časovne zahtevnosti priporočilnega sistema

Določitev predlog za nakupe

Določitev predlogov je predstavljena v prejšnjem poglavju v listingu 4.8. Metoda za določitev predlaganih izdelkov se sprehodi čez vsa naročila, ki so bila oddana po želenem datumu. Ob vsaki iteraciji se metoda sprehodi čez vse postavke naročila ter doda predlagane izdelke strankam.

Časovna zahtevnost te metode je tako $\sum_1^n \sum_1^m = \Theta(n * m)$, kjer

- n predstavlja število vseh naročil in
- m število vseh postavk v naročilu.

V praksi se izkaže, da je število postavk naročila sorazmerno majhno, največ do nekaj deset izdelkov. Ob tem opažanju lahko število postavk v naročilu omejimo s konstantno vrednostjo. Tako lahko število m obravnavamo kot konstantno. Časovna zahtevnost določanja predlaganih izdelkov se tako poenostavi na $\Theta(n)$.

Določitev priporočil za nakupe

Podobno kot določitev predlaganih izdelkov smo tudi izračun za priporočene izdelke predstavili v prejšnjem poglavju, in sicer v listingu 4.12. Metoda za določitev priporočenih izdelkov se sprehodi preko vseh naročil. Ob vsaki iteraciji pregleda vse postavke naročila in jih v zanki primerja z izdelki v seznamu priporočil, ki jih je definiral upravitelj.

Časovno zahtevnost metode za določitev priporočenih izdelkov lahko predstavimo z matematičnih izrazom $\sum_1^n \sum_1^m \sum_1^l = \Theta(n * m * l)$, kjer

- n predstavlja število vseh naročil,
- m je število vseh postavk naročila in
- l predstavlja število priporočil v seznamu priporočil.

Števili m in l sta v primerjavi s številom n majhni, poleg tega pa ju lahko prav tako obravnavamo kot s konstanto navzgor omejeni števili. Število postavk v naročilu redko kdaj preseže nekaj deset izdelkov. Tudi število priporočenih izdelkov, ki jih definira upravitelj, je majhno v primerjavi s številom naročil in v praksi ne preseže nekaj deset izdelkov. Tako lahko časovno zahtevnost določitve priporočenih izdelkov poenostavimo na $\Theta(n)$.

5.3 Nadaljnje delo in možne izboljšave

Idej za izboljšave je bilo med samo implementacijo spletne trgovine in priporočilnega sistema veliko. Odpira se veliko poti za optimizacijo in izboljšavo delovanja spletne trgovine. V tem razdelku podajamo samo nekaj najzanimivejših:

- **Ločen dostop za upravitelja trgovine:** Eno izmed možnih izboljšav vidimo pri dostopu do nadzorne plošče. Tu se lahko aplikacijo loči na spletno trgovino in nadzorno ploščo za upravitelja, ki bi obstajali na ločenih strežnikih. Tako se lahko dostop do nadzornega strežnika preko certifikata dovoli samo upravitelju.

- **Stalna integracija in avtomatski testi:** Spletna trgovina bi veliko pridobila z implementacijo stalne integracije (angl. continuous integration, CI), ki bi projekt spletne trgovine periodično zaganjala in na njej izvajala avtomatske teste, ki bi pokrivali večino funkcionalnosti.

- **Avtomatsko osveževanje kode:** Da bi bila koda vedno zadnje različice ter da razvijalcem ne bi bilo potrebno stalno posodabljati kode na produkcijskem strežniku, bi lahko skripta poskrbela za periodično povpraševanje in posodabljanje kode preko protokola Git.

- **Pametnejši algoritem določitve priporočil**: Trenutni algoritem za določitev priporočenih izdelkov je preprost. Algoritem vedno znova določi priporočila za izdelke, ki jih je že določil in se nahajajo v podatkovni bazi. Priporočilni sistem potrebuje boljšo strategijo, ki si bo

zapomnila zadnje pregledano naročilo in določevanje nadaljevala od zadnjega pregledanega naročila naprej.

- **Boljše proženje določitve priporočilnega sistema:** Spletna trgovina potrebuje pametnejšo strategijo proženja določitve predlaganih in priporočenih izdelkov. Trenutna rešitev, ki proži določitev vsak dan, ni optimalna.

- **Izboljšana določitev davčne stopnje:** Trenutni sistem trgovine določa davčno stopnjo globalno. Tako imajo vsi izdelki enako davčno stopnjo, ki je določena v nastavitvah. Izboljšava trenutnega sistema bi bila, da se davčna stopnja nastavlja glede na vrsto izdelka. Tako ima lahko spletna trgovina različne davčne stopnje glede na vrsto izdelka.

Poglavje 6

Sklepne ugotovitve

V sklopu diplomske naloge je bila implementirana spletna trgovina, ki premakne klasično nakupovanje in naročanje izdelkov na svetovni splet. S tem obiskovalcem ponudi možnost preprostejšega pregleda in naročanja izdelkov, hkrati pa upraviteljem trgovine omogoča poenoteno trgovanje in zmanjša stroške poslovanja. Ob implementaciji smo se spoznali z delovanjem spletnih trgovin in spoznali nekaj novih primerov dobre prakse implementacije in poslovanja spletnih trgovin.

Implementacija sledi modernim trendom spletnih trgovin, ki s privlačnim izgledom in prijetno uporabniško izkušnjo privabljajo obiskovalce in poskrbijo za enostaven proces naročanja izdelkov. Razvita trgovina vključuje večino osnovnih funkcionalnosti, ki jih ponujajo moderne spletne trgovine. V nadaljevanju smo uporabili nekatere naprednejše koncepte, ki spletno trgovino naredijo interaktivnejšo ter upraviteljem olajšajo delo pri vzdrževanju. Poskrbeli smo, da je postavitev spletne trgovine dobro dokumentirana in enostavna za postavitev v produkcijsko okolje. Posebno pozornost smo namenili varnosti, saj želimo, da je komunikacija uporabnika s spletno trgovino varna. Prav tako se želimo izogniti tveganju, da bi zaradi slabe varnosti spletne trgovine upravitelj naletel na kakršne koli nevšečnosti ali poslovno škodo. Z izbiro pravih tehnologij smo poskrbeli, da je nadgrajevanje sistema na novejše različice enostavno za upravitelja.

Na podlagi osnovnih konceptov izboljševanja prodaje smo spletno trgovino obogatili s priporočilnim sistemom. Trgovina je tako nadgrajena s predlogi izdelkov, ki bi stranko utegnili zanimati. Implementirali smo zaledni sistem, s katerim lahko upravitelj spletne trgovine nastavi parametre za prikaz priporočenih izdelkov strankam trgovine.

Med samo implementacijo smo dobili veliko idej za izboljšanje in optimizacijo spletne trgovine. Menimo, da je prostora za izboljšave veliko. Modularna implementacija trgovine omogoča, da izdelano rešitev uporabimo kot osnovo za postavitev morebitnih drugih spletnih trgovin.

Literatura

[1] Amazon.com™. Amazon.com™. Dosegljivo: http://www.amazon.com, 2016. [Dostopano: 8. 6. 2016].

[2] Mitchell Anicas. How to secure nginx with let's encrypt on ubuntu 16.04. Dosegljivo: https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-16-04, 2016. [Dostopano: 10.10.2016].

[3] DealExtreme™. Dealextreme™. Dosegljivo: http://www.dx.com, 2016. [Dostopano: 8. 6. 2016].

[4] Podjetje eBay™. Podjetje ebay™. Dosegljivo: http://www.ebay.com, 2016. [Dostopano: 8. 6. 2016].

[5] Django Software Foundation. Varnost v ogrodju django. Dosegljivo: https://docs.djangoproject.com/en/1.10/topics/security/, 2016. [Dostopano: 26. 9. 2016].

[6] Internet Security Research Group. Let's encrypt. Dosegljivo: https://letsencrypt.org/about/, 2016. [Dostopano: 18.10.2016].

[7] Definicija jezika HTML. w3.org. Dosegljivo: https://www.w3.org/TR/REC-html40-971218/conform.html, 2016. [Dostopano: 13.9.2016].

[8] nginx.com. Strežnik nginx. Dosegljivo: https://http://nginx.com/, 2016. [Dostopano: 12.10.2016].

[9] Django Oscar. Dokumentacija oscar. Dosegljivo: `http://django-oscar.readthedocs.io`, 2016. [Dostopano: 30. 9. 2016].

[10] Django Oscar. spletna trgovina oscar. Dosegljivo: `https://github.com/django-oscar/django-oscar`, 2016. [Dostopano: 30. 9. 2016].

[11] B Joseph Pine. Mass customization: the new frontier in business competition. 1993.

[12] Frederick F Reichheld. Loyalty-based management. *Harvard business review*, 71(2):64–73, 1992.

[13] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. *Recommender systems handbook*, pages 1–35, 2011.

[14] J Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166. ACM, 1999.

[15] J Ben Schafer, Joseph A Konstan, and John Riedl. E-commerce recommendation applications. *Applications of Data Mining to Electronic Commerce*, pages 115–153, 2001.

[16] wikipedia.com. Protokol wsgi. Dosegljivo: `https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface`, 2016. [Dostopano: 12.10.2016].